3D position estimation of sports players through multi-view tracking

by

Robbie Vos

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science

at

Stellenbosch University

Department of Mathematical Sciences

Faculty of Science

Supervisor: Dr Willie Brink
Co-supervisor: Prof. Ben Herbst

Date: October 2010

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 1 October 2010

Abstract

Extracting data from video streams and using the data to better understand the observed world allows many systems to automatically perform tasks that ordinarily needed to be completed by humans. One such problem with a wide range of applications is that of detecting and tracking people in a video sequence. This thesis looks specifically at the problem of estimating the positions of players on a sports field, as observed by a multi-view camera setup.

Previous attempts at solving the problem are discussed, after which the problem is broken down into three stages: detection, 2D tracking and 3D position estimation. Possible solutions to each of the problems are discussed and compared to one another.

Motion detection is found to be a fast and effective solution to the problem of detecting players in a single view. Tracking players in 2D image coordinates is performed by implementing a hierarchical approach to the particle filter. The hierarchical approach is chosen as it improves the computational complexity without compromising on accuracy. Finally 3D position estimation is done by multi-view, forward projection triangulation. The components are combined to form a full system that is able to find and locate players on a sports field.

The overall system that is developed is able to detect, track and triangulate player positions. The components are tested individually and found to perform well. By combining the components and introducing feedback between them the results of the individual components as well as those of the overall system are improved.

Opsomming

Deur data uit 'n video-stroom te ontrek, en die data te gebruik om die wêreld wat waargeneem word beter te verstaan, kan baie rekenaarstelsels take outomaties voltooi wat voorheen deur 'n mens sou gedoen moes word. Een so 'n probleem wat 'n wye toepassingsveld het, is om mense te vind en te volg in 'n video. Hierdie tesis kyk spesifiek daarna om die posisie van spelers op 'n sportveld te vind, gegee 'n klomp kameras wat na die veld kyk.

Daar word na vorige stelsels wat hierdie probleem probeer oplos gekyk, waarna die probleem in drie dele opgedeel word: vind die spelers, volg die spelers in 2D en skat die posisie van die spelers in 3D. Moontlike oplossings vir elk van hierdie dele word bespreek en vergelyk met mekaar.

Daar word gevind dat om beweging te identifiseer 'n eenvoudige manier is om die spelers te vind. Hulle word dan gevolg in 2D beeldkoördinate deur gebruik te maak van 'n hiërargiese implementasie van die partikel-filter. Die hiërargiese implementering word gekies omdat dit die spoed van die partikel-filter verbeter, sonder om die akkuraatheid te verswak. Laastens word die 3D posisie gevind deur multi-sigpunt, voorwaartse projeksie triangulering. Die verskillende komponente word kombineer om 'n volledige stelsel te vorm wat spelers kan vind en plaas op 'n veld.

Die volledige stelsel wat ontwikkel is, is in staat om spelers te vind, volg en hulle posisies te bepaal. Elk van die individuele komponente word getoets, en daar word gevind dat hulle goed op hulle eie werk. Deur die komponente te kombineer en terugvoer tussen verskillende komponente te bewerkstellig word die resultate van die individuele komponente, sowel as dié van die volledige stelsel nog verbeter.

Acknowledgments

I would like to thank the following people for their aid and support during the course of my studies:

- My supervisor, Willie Brink, for going far beyond the call of duty.
- My co-supervisor, Prof. Ben Herbst, for his insights and guidance.
- MIH for all their financial support and inspiration.
- The NRF for financial support.
- My girlfriend, Mariaan Smit, for all her love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer vision is the field of study relating to machines that observe the world around them. As a scientific discipline it is concerned with the technology of artificial systems extracting information from images or sequences of images. Images may come from a variety of sources that may include single snap-shot cameras, video cameras or multiple synchronized cameras.

Applications for computer vision range greatly between various fields. Security is one field that uses computer vision to a great extent. Video motion detection allows for automatic intruder alerts while more advanced systems are able to detect suspicious people or parcels in public areas. Systems that are able to measure the length of a queue or patterns in human movement are used in shopping malls and airports to optimize personnel and layout decisions. In a sports environment video data can be analyzed to extract statistical information such as how often a person handled the ball or which side had more possession.

This thesis considers the problem of tracking the 3D positions of players moving about on a sports field. This problem requires one to use techniques from several areas of computer vision. Combining the various techniques in a computationally effective manner poses a challenge and reaching a real-time implementation is not a trivial problem.

Figure 1.1: Examples of computer vision used in different fields. From left to right: motion detection, sports statistics and pedestrian flow.

1.1 Background and Motivation

As technology has improved over the past years, the number of people following sports around the world has increased correspondingly. Before the invention of radio the only way to gain knowledge about events from far away was by word of mouth or reading a newspaper or magazine. With the arrival of radio people could listen to live commentary as the game was unfolding. Later when television became available it became possible to watch a game happening on the other side of the world. With every step forward in technology it became easier to follow sports and allowed larger audiences to follow the action.

With the greater following that sports obtained, the analysis of how teams and players perform during matches and seasons also gained interest. Spectators and fans are increasingly looking for up-to-date statistics on all aspects of their game of choice. Much of these statistics are manually extracted while watching the game or from video footage after the game has completed.

A system that is able to track players on a field during a game will be able to automatically provide a range of statistics that is relevant to analyzing player and team performances. It will be possible to calculate how much distance players are covering in a match as well as which areas of the field they spend the majority of time. This can be used to measure the work rate of different players on the field or to compare tactical strategies between teams.

The use of cameras in developing such a system will also provide various advantages above, say, attaching a GPS or other tracking device to each player. One of the big advantages is that the use

Figure 1.2: Viewing of sport events as technology progressed.

of cameras is non-intrusive. Players will not need to attach a device to their clothing or person. In contact sports such tracking devices also run the risk of being damaged during physical contact.

The costs involved when using cameras also make them an attractive option. Although setup costs may be high if high-quality cameras are used, these costs are incurred only once. Using personal devices for each player would require continuous costs to maintain such devices. A final advantage is that cameras are a passive medium. A solution using radio or radar waves might work in a similar fashion to a camera-based solution, however it would need to project those waves onto the field. This may interfere with transmission of audio and video feeds to viewers around the world.

1.2 Problem Statement

Given the world-wide interest in sport and sport statistics, along with the numerous advances in technology over the past decades, the problem that this study will address is: ***Tracking the 3D positions of players on a sports field using multiple stationary cameras.***

The following section expands on the problem statement and includes specific aims and objectives of the study.

1.3 Aims and Objectives

The primary aim of this study is to design and implement a computer vision system that is able to track players as they move around on a sports field. Different components are compared based

on accuracy and computational complexity to find a suitable compromise. The components are then combined into a complete system that is able to detect and track players moving about on a field, as observed by several cameras. Attempts must be made to have the system run in real-time (approximately 25 frames per second) to allow relevant statistics to be gained during the playing of a match. The work in this thesis is limited to detecting and tracking players in non-contact sports such as field hockey and soccer, due to the increased difficulties that arise from re-identifying players after complex multi-person contact situations.

The setup of such a system, along with the calibration of the cameras, is a vital component of the whole. If this is not done properly all the data that the system extracts will be inaccurate. As a result the setup must be accurate and easy to implement by non-experts. It should also be easy to modify the setup at a later stage.

After the setup procedure the next stage of importance is the computation that needs to be performed on each camera stream individually. For each camera stream, the system must be able to detect players that are within the camera's field of view and, having detected them, track those players through the video sequence. The detection step is important, without it the system will not be able to track any players. On the other hand new players do not enter the field of view very regularly. As such the detection stage must be computationally inexpensive (as not to waste computational power on redundant searches), while still being able to detect new players within a reasonable time (say around 10 frames). The detection stage should also be robust against false positives. The tracking of players in each video sequence forms the largest component of the system and it requires a high degree of accuracy. It is of vital importance that the tracker does not lose any players it is tracking as this may have a severely negative influence on the 3D tracking accuracy.

The final component of the tracker is the 3D position estimation of players on the field. Combining the tracking data of each of the individual cameras, this stage must be able to accurately estimate the position of each player on the field. The triangulation procedure must be computationally inexpensive to increase the achievable frame-rate of the system.

An additional aim of the thesis is to implement the modules required for the various stages of the project, developing the code rather than using pre-developed modules. This allows the code that is written to be developed specifically for the given application. Writing the code oneself also gives

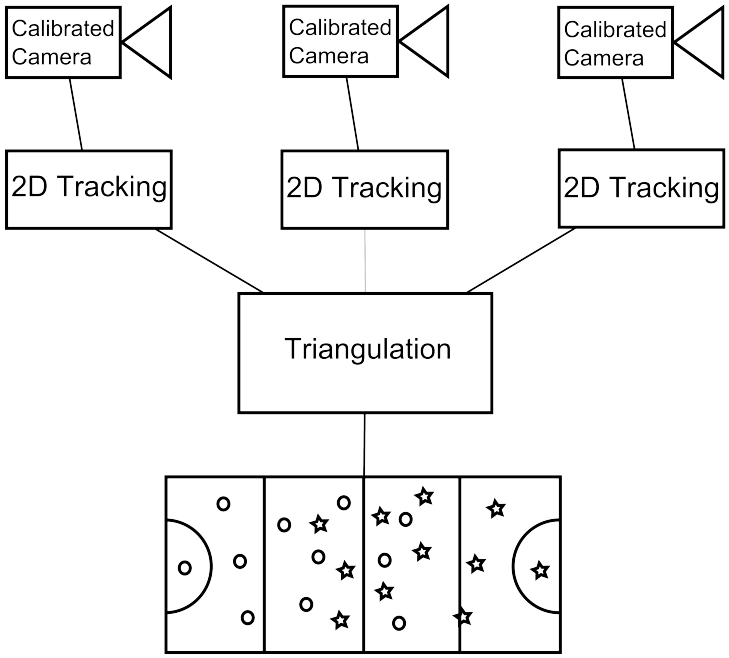


Figure 1.3: Overview of the tracking system.

one complete control over the code, so that modifications or improvements can easily be made.

1.4 System Overview

The system that is developed through this study can be broken down into several component parts. The three core components of the system are: camera setup, 2D tracking and 3D tracking as illustrated in figure 1.3.

The camera setup component relates to the physical setup that would be made at a field where the system would be used. Along with the physical setup of the cameras the calibration of the cameras to the world around them needs to be calculated. Only by accurately fixing the internal parameters as well as the position of a camera in the world can it be used to perform measurements on the world that it observes.

The second component is the processing that needs to be performed on each video stream individually. The first step is to detect players that enter the field of play. A motion detection solution is implemented to detect players moving around in the field. Once players have been detected they

can then be tracked through a video sequence. To accomplish this several different approaches are considered, and a hierarchical approach to the particle filter is implemented.

The final component of the system combines the tracking data from each of the individual cameras to determine the 3D positions of players on the field. Corresponding players are found between the different views, and their location on the field is then triangulated from the multiple views.

1.5 Thesis Outline

In the rest of the thesis different areas of interest pertaining to the problem are discussed. In chapter 2 some previous attempts at solving the problem are discussed. The different approaches are compared based on their implementation, accuracy and computational complexity.

In chapter 3 the pinhole camera model is described. It is then shown how to calibrate (calculate the internal and external parameters of) a single or set of cameras. The final section of the chapter describes how the camera model is used to triangulate the real-world coordinates of a point visible in multiple cameras.

Chapter 4 looks at the problem of detecting the presence of people in an image or video sequence. The first part of the chapter discusses various algorithms for detecting and locating people in an image, while the second part concentrates on using motion detection to locate people moving through a video sequence.

The problem of tracking people through a video sequence is covered in chapter 5. Two categories of tracking are discussed. The first tracks people through the sequence by detecting them in each frame. The second category uses a filter to estimate the position and then compares the estimate to some model of what is expected, to produce a more accurate result.

In chapter 6 the various components discussed in the previous chapters are combined to form a solution to the stated problem. Details for camera calibration, 2D tracking and 3D triangulation are given.

The results obtained by the system are presented and discussed in chapter 7. Individual components are tested, followed by tests of the complete system.

The final chapter, chapter 8, presents some conclusions based on the problem statement, objectives and obtained results. Some recommendations are made for future research to expand and improve the system.

Chapter 2

Related Work

Before one begins to develop a system it is important to investigate similar systems that have been produced or proposed by other researchers. Such systems can be compared in three areas: methodology, accuracy and speed performance.

Different approaches need to be compared to find similarities and differences. By doing this it allows one to find positive and negative trends in past research. Research of past work may also provide one with innovative ideas used by different researchers that may be combined to produce a more accurate or efficient system.

For the rest of this chapter various such previous approaches are summarized and then discussed according to their strengths and weaknesses. Single-view approaches are compared first, followed by multi-view solutions.

2.1 Single View

Single view refers to systems that attempt to solve the problem using video captured by a single camera.

2.1.1 Condensation Tracking

In the work by Needham et al. [39] players in a five-a-side soccer match are tracked using a single static camera that has a view of the entire field.

To detect players on the field, colour models of foreground and background regions are created offline, prior to running the tracker. The regions are manually identified from a set of frames in the sequence. When running the tracker on the sequence each image is then segmented into foreground and background regions and a bounding box is placed around each foreground region.

The players on the field are tracked using the Condensation algorithm [30]. Each player is represented by a bounding box that is described by four parameters: the x and y location of the center of the bottom of the bounding box and the width and height of the bounding box.

The x and y parameters of the bounding box are used to calculate the location of the player on the field by the assumption that those parameters correspond to the location of the feet of the player on the field. By calibrating the camera to the ground plane (field) the intersection between a line extended from the camera center to the bounding box point and the field can be calculated.

At each frame, estimates of player positions are matched to the extracted foreground regions and a best match is found, after which the player bounding boxes are updated. The estimates of player positions are improved by using Kalman filtering in addition to the Condensation algorithm.

This method of player tracking provides some good results for the chosen application but it does have some severe limitations. Offline foreground and background parametrization implies that the method cannot be run in real-time or without human interaction. Perspective distortion from the camera location also causes the 3D position estimation to be inaccurate on the far side of the field.

2.1.2 Tracking by Detection of Shirt and Pants Regions

The problem of tracking players on a field from a single pan-tilt-zoom (PTZ) camera is tackled by Yamada et al. [55]. Using a moving camera presents new problems, especially in the calibration.

Since the camera parameters can change between each frame the camera needs to be calibrated

at each frame. Lines and curves on the field are found in each frame and compared to a model of a soccer field. By comparing the detected lines and curves with the model the camera parameters can be calculated at each frame.

The detection of players is done by identifying shirt and pants regions in each image. The pants and shirt regions are then matched by checking for regions that align vertically.

Player locations on the field are found in the same way as Needham et al. [39] by finding the position of feet on the field using the intersection of a line through the bottom of the player region and the ground plane.

Players are tracked through a sequence by estimating their position in each frame by simple linear extrapolation between frames. This is possible as player motion can be approximated as near constant over short periods of time. The estimated position of each player is projected back onto the image plane and a region around the backprojected point is searched for a pants-shirt match.

Although this solution does successfully track players through the difficulties of a moving camera, it does fail in some important areas. The greatest of these is that it is unable to handle occlusions. When occlusions occur the occluded players are tracked as a single entity until the occlusion passes.

2.1.3 Template Matching with Kalman Filtering

Another single-view tracking attempt was done by Choi et al. [12], also using a single PTZ camera. In this solution the calibration is done by creating a model of the field. The view of the field is then matched to this model and the transformation between the view and the model can be calculated.

To detect players in the image the field area in the image is first found. A mask of the field is created by identifying the primary colour in the image (the field occupies the largest part of the image). Players are then found by masking the image with the field mask. Regions where players are present are then identified by looking for areas where the image differs from the mask.

The tracking of players is done by building a shape template of each player. A Kalman filter is used to predict the position of a player in a frame. A region around the predicted point is then searched for a match of the player template. The best match is used as the measurement step of the

Kalman filter, and the template is updated.

Player positions on the field are then found by using the transformation between the field view and the model. The player region is transformed onto the field model, giving the location of the player on the field.

This solution is able to follow players through occlusions as long as those occlusions occur between players of opposing teams, by comparing colours of the uniforms. Occlusions between players of the same team cannot be handled.

Single-view tracking solutions all share several problems. The first problem is that with only one camera a trade-off needs to be made between field-of-view and resolution. When viewing the entire field the resolution of far-away areas will be very low causing many missed detections. When viewing only a section of the field some players will be outside the field of view and will not be tracked. Single view solutions are also unable to triangulate player locations accurately through occlusions.

In the next section some multi-view tracking solutions are discussed. These multi-view solutions are able to deal with the problems that the single-view solutions have.

2.2 Multi-View

Many of the problems that single view systems have can be overcome by using multiple cameras. First some systems that are able to match people between multiple views and track them in 2D are discussed, after which full systems that are able to track people in 3D are covered.

2.2.1 2D Multi-View Tracking

Khan et al. [32] tackled the problem of tracking people moving through a building using multiple views. By finding the field-of-view lines for each camera they are able to identify people as they enter the view of a new camera by comparing what view lines the person crossed in other cameras they are being tracked in. By analyzing the movement of people when they enter the view of a new camera the field-of-view lines of the cameras are also able to be calculated automatically.

Another method to track people in multiple views is done by Cai et al. [8]. People are detected by segmenting the image into foreground and background, by building a model of the background and comparing the current view to the model. Foreground regions are then broken into bounding boxes using a window slicing technique [31]. The foreground regions are then analyzed to detect human shapes. A 2D model of the human body is chosen by combining a set of rectangles and ellipses.

People are tracked through a video sequence by selecting a number of feature points and recording the geometric relation between the points. In successive frames detected people are compared to previous sets of feature points to find a match for people between frames. Matches between cameras are found by comparing the positions and velocities of people through the video sequence.

Whilst both of these systems are able to track people in 2D and perform matches between the cameras they lack some functionality that is required for 3D tracking. The biggest problem is that neither of the two solutions are able to calculate the 3D position of a tracked person. Another problem with the two systems is that they are unable to track people through occlusions.

2.2.2 Multi-View 3D Tracking

The first of the full multi-view 3D tracking systems was developed by Alahi et al. [2]. In their system players on a basketball field are tracked using a selection of planar and omnidirectional cameras.

Using adaptive mixture models from [47] a foreground image is extracted for each camera. Having calibrated the cameras beforehand the foreground silhouettes for each camera can be projected onto the court ground plane. By matching the ground plane projections of each camera and modelling the player behaviour using the work of [3] players can be tracked on the court through a sequence.

This tracking solution is able to track players in a video sequence but achieves low results for precision and recall (precision: 76% and recall: 72%). The tracking approach is also rather simple and is unable to track players through occlusions.

Another multi-view 3D tracking technique was developed by Xu et al. [54]. A mask of the field is extracted using background modelling techniques. This mask is applied to all subsequent images

to limit detection and tracking only to regions of the image that correspond to the field of play. Players are then found on the field using the mixture of Gaussians approach (as described in section 4.2.2 of this thesis).

2D player tracking is performed using a Kalman filter. For each player in each view a Kalman filter is created to track that player through the video sequence. Measurement updates are taken from the motion detection using a nearest neighbour approach.

For 3D tracking a Kalman filter is again used. Player bounding boxes from the 2D stage are projected onto the field of play and used as measurements for the 3D Kalman tracking filters. The system assumes 25 people are on the field at all times: 11 players on each team, a referee and two linesmen. The bounding box projections are evaluated using several criteria and are then assigned to each of the expected 25 people to update the Kalman filter.

This proposed solution provides good results for tracking the 25 players over long video sequences while remaining computationally inexpensive. The system does, however, fail at some important aspects. By forcing the system to use 25 people it may provide false positives in cases where there are less players, e.g. a player is off due to injury, or fail to detect people when there are more than 25 people on the field, e.g. medical staff enter the field causing the system to track them and ignore players. Players are also often triangulated using a single camera which may provide inaccurate position results.

Having introduced the problem in chapter 1 and discussed previous attempts to solve the problem in this chapter the rest of this thesis discusses a proposed solution to the problem. The next few chapters discuss the various elements that need to be combined to provide a full solution, and are then combined in chapter 6.

Chapter 3

The Geometry of Cameras

Before one can work with a camera and perform calculations from a video sequence it is important to establish a mathematical model describing a camera. Once this is understood it becomes possible to use the data contained in the video sequence to calculate positions of real-world features. In this chapter the pinhole camera model is first described and then a method for calculating the camera parameters is given. Next the relationship between multiple cameras capturing the same scene is explored and finally triangulating real-world features from several views is explored.

3.1 Single Camera Model

The pinhole model, shown in figure 3.1, is used to describe the geometry of a camera. In this model the camera is defined by the camera center, $\mathbf{C}$, and the image plane. Any point $\mathbf{X}$ in $\mathbb{R}^3$ can be projected onto the image plane by tracing a ray from $\mathbf{X}$ to $\mathbf{C}$. The point $\mathbf{x}$ where the ray intersects the image plane becomes the projected point. Note that $\mathbf{X}$ is a point in $\mathbb{R}^3$ whilst $\mathbf{x}$ is a point in $\mathbb{R}^2$ on the image plane.

To represent this projection mathematically the points $\mathbf{X}$ and $\mathbf{x}$ need to be transformed into the homogeneous coordinate system. The camera matrix $\mathbf{P}$ can now be introduced as the operator that

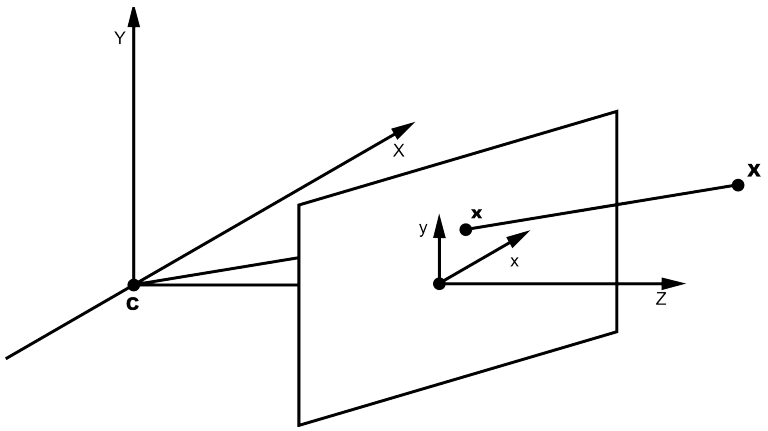


Figure 3.1: Projection of a $3D$ point $\mathbf{X}$ onto the image plane using a pinhole camera model.

maps $\mathbf{X}$ to $\mathbf{x}$:

$$\mathbf{x} = \mathbf{PX}. \tag{3.1}$$

In homogeneous coordinates $\mathbf{X}$ is a 4×1 vector and $\mathbf{x}$ is a 3×1 vector implying that $\mathbf{P}$ is a 3×4 matrix.

We now proceed to analyze the $\mathbf{P}$ matrix, starting with simple projection and building it up as we introduce more factors until we finally arrive at the full definition of $\mathbf{P}$. The first step is to look at a simple projection of a point onto the image plane. In figure 3.2 the camera center is at the origin and the image plane is parallel to the x-y plane.

From figure 3.2 it is clear that $\mathbf{X}$ projects onto $\mathbf{x}$. Using similar triangles we can show that this projection takes the point $(X, Y, Z)^T$ to $(fX/Z, fY/Z, f)^T$, where f is the focal length of the camera, with both points in $\mathbb{R}^3$ and in Euclidean coordinates. The second point, however, now lies on the image plane. This allows us to write it as $(fX/Z, fY/Z)^T$ in image coordinates (all points on the image plane will have a z component of f). In homogeneous coordinates the second point becomes $(fX/Z, fY/Z, 1)^T$ which can be rewritten as $(fX, fY, Z)^T$. This projection can now be described by expressing (3.1) as

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.2}$$

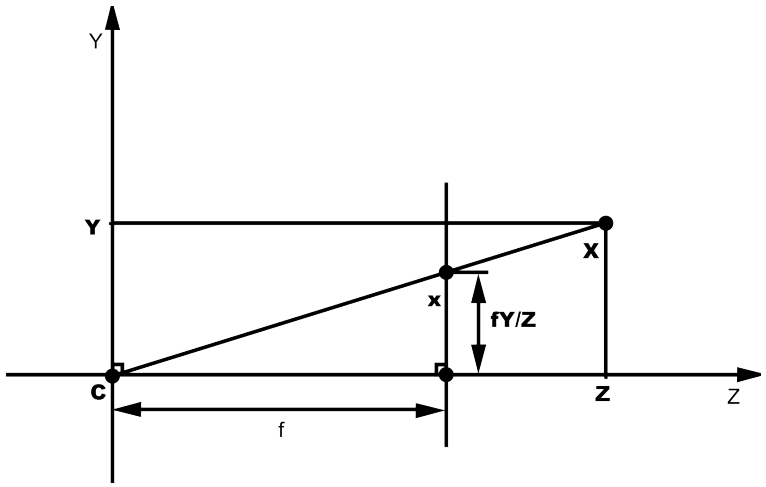


Figure 3.2: Simple projection using camera coordinates in the z-y plane, where f is the focal length.

Next we look at the influence of the image coordinate system. To do this the principal point first needs to be defined. Extending a line from the camera center perpendicular to the image plane provides one with two basic features. The line forms the principal axis, and the point where it intersects the image plane is the principal point. $\mathbf{P}$ as described above assumes that the origin of the image coordinates is at the principal point $(0, 0, f)^T$. If this is not the case, we need to add the x and y coordinates of the camera center to the projected point $\mathbf{x}$. This gives us the projection:

$$(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T. \tag{3.3}$$

Moving to homogeneous coordinates again gives us

$$x = (fX/Z + p_x, fY/Z + p_y, 1) = (fX + Zp_x, fY + Zp_y, Z). \tag{3.4}$$

This can be represented in $\mathbf{P}$ by including the p_x and p_y components:

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.5}$$

Having now looked at the geometry of an ideal pinhole camera we need to introduce two factors that occur in real-world cameras. The first of these is the ratio between the unit length (or pixel

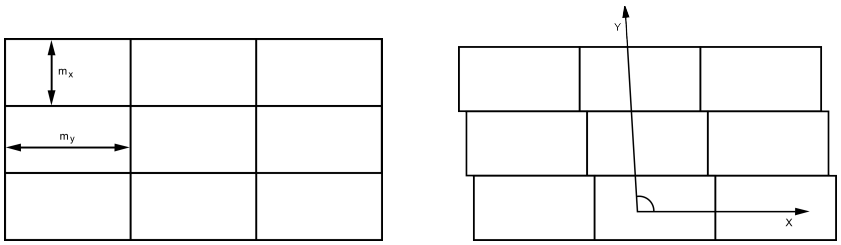


Figure 3.3: Illustration of the camera axis ratio (left) and camera skew parameter (right).

size) in the direction of the two axes. Up to now we have assumed that this ratio is 1, as would result from perfectly square pixels. In real-world cameras this is often not the case and pixels actually become rectangular as shown in figure 3.3 (a). To account for this the camera matrix needs to be multiplied by $\mathrm{diag}(m_x, m_y, 1)$ from the left, with m_x proportional to the lenght in the x direction and m_y proportional to the lenght in the y direction, giving:

$$\mathbf{P} = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.6}$$

where $(\alpha_x, \alpha_y) = (m_x f, m_y f)$ and $(x_0, y_0) = (m_x p_x, m_y p_y)$.

The second factor we need to consider is called the skew factor. This arises if the x and y axes of the camera are not perfectly perpendicular, as shown in figure 3.3 (b). To account for this the parameter s is included into the $\mathbf{P}$ matrix giving us:

$$\mathbf{P} = \begin{bmatrix} \alpha_x & s & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.7}$$

Up to this point the $\mathbf{P}$ matrix contains all the parameters related to the internal structure of the camera. We now need to look at the external parameters relating the camera position to the real-world coordinate system. Before we continue it is useful at this point to rewrite the $\mathbf{P}$ matrix as

$$\mathbf{P} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]. \tag{3.8}$$

$\mathbf{K}$ is called the camera calibration matrix and contains all the internal parameters, i.e.

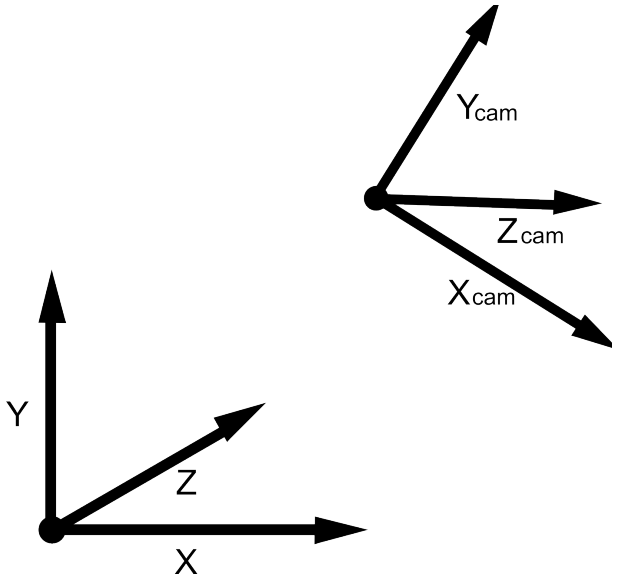


Figure 3.4: Relation between world and camera coordinate systems.

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.9}$$

The $\mathbf{P}$ matrix in (3.7) is sufficient to project any point in $\mathbb{R}^3$ onto the image plane, assuming the point is given in the camera coordinate system. It is often the case that the camera, $\mathbf{X}_{cam}$, and real-world, $\mathbf{X}$, coordinates of the point do not agree. It becomes necessary to relate the two coordinate systems to one another. The two systems are related by a rotation and a translation as shown in figure 3.4. A simple relationship exists, and can be written as:

$$\tilde{\mathbf{X}}_{cam} = \mathbf{R}\left(\tilde{\mathbf{X}} - \tilde{\mathbf{C}}\right), \tag{3.10}$$

where $\tilde{\mathbf{C}}$ is the 3×1 vector for the camera center in (Euclidean) world coordinates and $\mathbf{R}$ is the 3×3 rotation matrix between the two coordinate systems. The tilde indicates Euclidean coordinates, i.e.

$$\mathbf{X} = \begin{pmatrix} \tilde{\mathbf{X}} \\ 1 \end{pmatrix}. \tag{3.11}$$

Equation (3.10) can be rewritten in homogeneous coordinates as

$$\mathbf{X}_{cam} = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ \mathbf{0} & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.12}$$

We can combine equations (3.8) and (3.12) to arrive at

$$\mathbf{P} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right]. \tag{3.13}$$

Equation (3.13) now contains all the parameters needed to project any point onto the image plane and the final projection equation becomes:

$$\mathbf{x} = \mathbf{K}\mathbf{R}\left[\mathbf{I} \mid -\tilde{\mathbf{C}}\right]\mathbf{X}. \tag{3.14}$$

3.2 Single Camera Calibration

In the previous section the pinhole camera model was described. In this section the process of calculating $\mathbf{P}$ is described. The $\mathbf{P}$ matrix from (3.13) has 11 degrees of freedom: 3 for the rotation matrix $\mathbf{R}$, 3 for the translation vector $\tilde{\mathbf{C}}$ and 5 for the calibration matrix $\mathbf{K}$.

To solve for all the parameters 11 linearly independent equations are required. The equations can be obtained from point correspondences $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$ where $\mathbf{x}_i = \mathbf{P}\mathbf{X}_i$ for every i. Equations are derived by setting the vector cross product $\mathbf{x}_i \times \mathbf{P}\mathbf{X}_i$ equal to $\mathbf{0}$ since any two parallel vectors in homogeneous coordinates refer to the same point in the image plane. Writing $\mathbf{P}$ as a vector of its rows, $\mathbf{P}\mathbf{X}_i$ becomes:

$$\mathbf{P}\mathbf{X}_i = \begin{pmatrix} \mathbf{p}^{1T}\mathbf{X}_i \\ \mathbf{p}^{2T}\mathbf{X}_i \\ \mathbf{p}^{3T}\mathbf{X}_i \end{pmatrix}, \tag{3.15}$$

where $\mathbf{p}^{jT}$ denotes the jth row. The cross product can be expressed as the product of a skew-

symmetric matrix and a vector, leading to the following equations

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0}, \tag{3.16}$$

where $\mathbf{X}_i = (x_i, y_i, w_i)$ and $\mathbf{p} = \left(\mathbf{p}^{1T},\ \mathbf{p}^{2T},\ \mathbf{p}^{3T}\right)^T$ is a 12×1 vector containing all the elements of $\mathbf{P}$. From(3.16) it would appear that each point correspondence provides three equations. However, this is not the case as only two of the equations are linearly independent causing (3.16) to reduce to

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{bmatrix} \begin{pmatrix} \mathbf{p}^1 \\ \mathbf{p}^2 \\ \mathbf{p}^3 \end{pmatrix} = \mathbf{0}. \tag{3.17}$$

Each point correspondence provides us with two linear equations. To solve the eleven degrees of freedom we need eleven equations, or at least $\lceil \frac{11}{2} \rceil = 6$ point correspondences. It is important to note that no three points chosen should be collinear as they will not provide linear independent equations. Stacking all the point correspondence equations into a single matrix, $\mathbf{A}$, the system $\mathbf{Ap} = \mathbf{0}$ can be solved by solving for the right null space of $\mathbf{A}$ using e.g. singular value decomposition (see Appendix B).

Due to the presence of noise and the possibility of measuring errors $\mathbf{Ap} = \mathbf{0}$ will often not have a non-trivial solution, and some approximation of the null space will need to be made, e.g. using a least squares approach. To increase the accuracy of such an approximation it is desirable to increase the number of equations used by increasing the number of point correspondences.

Having now calculated the $\mathbf{P}$ matrix it is useful to decompose it into the $\mathbf{K}$, $\mathbf{R}$ and $\tilde{\mathbf{C}}$ components. The aim is to find an upper-triangular matrix $\mathbf{K}$, a rotation matrix $\mathbf{R}$ and a 3×1 column vector $\tilde{\mathbf{C}}$ such that $\mathbf{P} = \mathbf{KR}\left[\mathbf{I}| - \tilde{\mathbf{C}}\right]$.

To this end, let $\mathbf{P}_{1:3}$ be the first 3 columns of $\mathbf{P}$, and $\mathbf{p}_4$ the 4^{th} column. Let

$$\mathbf{W} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{3.18}$$

note that $\mathbf{W}^{-1} = \mathbf{W}^T = \mathbf{W}$, and perform QR-factorization on $(\mathbf{WP}_{1:3})^T$, such that $(\mathbf{WP}_{1:3})^T = \widehat{\mathbf{Q}}\widehat{\mathbf{R}}$ with $\widehat{\mathbf{Q}}$ orthogonal and $\widehat{\mathbf{R}}$ upper-triangular.

The following set of matrices can now be calculated:

$$\begin{aligned} \mathbf{K} &= \mathbf{W}\widehat{\mathbf{R}}^T\mathbf{W} \\ \mathbf{R} &= \mathbf{W}\widehat{\mathbf{Q}}^T \\ \tilde{\mathbf{C}} &= -\mathbf{R}^T\mathbf{K}^{-1}\mathbf{p}_4 \end{aligned} \tag{3.19}$$

Note that $\widehat{\mathbf{R}}^T$ is a lower-left-triangular matrix, hence $\widehat{\mathbf{R}}^T\mathbf{W}$ is lower-right-triangular, and $\mathbf{K} = \mathbf{W}\widehat{\mathbf{R}}^T\mathbf{W}$ is an upper-right-triangular matrix. Also, $\widehat{\mathbf{Q}}$ is orthogonal, hence $\widehat{\mathbf{Q}}^T$ and $\mathbf{R} = \mathbf{W}\widehat{\mathbf{Q}}^T$ are also orthogonal.

It remains to show that (3.19) does indeed produce a decomposition of $\mathbf{P}$ in the form $\mathbf{P} = \mathbf{KR}\left[\mathbf{I}|-\tilde{\mathbf{C}}\right]$. From (3.19),

$$\mathbf{KR} = \mathbf{W}\widehat{\mathbf{R}}^T\mathbf{W}\mathbf{W}\widehat{\mathbf{Q}}^T = \mathbf{W}\widehat{\mathbf{R}}^T\widehat{\mathbf{Q}}^T = \mathbf{W}\left(\widehat{\mathbf{Q}}\widehat{\mathbf{R}}\right)^T = \mathbf{W}\left[\left(\mathbf{WP}_{1:3}\right)^T\right]^T = \mathbf{WWP}_{1:3} = \mathbf{P}_{1:3}, \tag{3.20}$$

and

$$-\mathbf{KR}\tilde{\mathbf{C}} = -\mathbf{KR}\left(-\mathbf{R}^T\mathbf{K}^{-1}\right)\mathbf{p}_4 = \mathbf{KR}\mathbf{R}^T\mathbf{K}^{-1}\mathbf{p}_4 = \mathbf{K}\mathbf{K}^{-1}\mathbf{p}_4 = \mathbf{p}_4. \tag{3.21}$$

Therefore $\mathbf{KR}\left[\mathbf{I}|-\tilde{\mathbf{C}}\right] = \mathbf{P}$.

Note that QR-factorization is unique only up to sign, and implies that any column of $\mathbf{K}$ and corresponding row of $\mathbf{R}$ can be multiplied by -1, and leave the product $\mathbf{KR}$ unchanged.

Luckily we can remove this ambiguity be noting two properties. First, the focal length of the camera must be positive, hence the first two entries on the diagonal of $\mathbf{K}$ must be positive. Second, the orthogonal matrix $\mathbf{R}$ must be a pure rotation (not reflection) and hence $\det(\mathbf{R})$ must be 1 (not -1). Using these restrictions the component matrices of $\mathbf{P}$ can be found uniquely.

3.3 Multiple Camera Calibration

The internal parameters of a camera remain constant over time (zooming may change the focal length, but it is assumed to remain constant). The external parameters, however, may change as

the camera moves around. For accurate triangulation (as discussed later) it is important that these external parameters are calculated accurately once every camera has been fixed at its location.

3.3.1 Absolute Calibration

One method to accomplish multiple camera calibration is to calibrate each camera individually against the same real-world coordinate system. With the internal parameters for each camera known there are only six degrees of freedom that remain for each camera: three for rotation and three for translation. The six degrees of freedom can be solved using a minimum of three point correspondences for each camera and is known as the three point relative pose problem.

Several methods for solving the three point relative pose problem have been presented in the literature. The first solution was presented by Grunert [23] in 1841 while more recent solutions have been presented, amongst others, by Grafarend et al. [22], Tsai [48] and Mozerov et al. [38]. A simple linear solution to the three point relative pose problem is presented in [25] and restated here.

To restate the problem, given three points in the 3D camera coordinate system and their corresponding three points in the 3D world coordinate system one wants to determine the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ that satisfies

$$\mathbf{p}_i = \mathbf{R}\mathbf{p}'_i + \mathbf{t}, \qquad i = 1, 2, 3, \tag{3.22}$$

where $\mathbf{p}_i = (x_i, y_i, z_i)^T,\ i = 1, 2, 3$ are the points in the 3D world coordinate system and $\mathbf{p}'_i = (x'_i, y'_i, z'_i),\ i = 1, 2, 3$ the points in the 3D camera coordinate system. $\mathbf{R}$ is a 3×3 orthogonal matrix such that $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $\mathbf{t} = (t_x, t_y, t_z)^T$.

To arrive at a linear solution the $\mathbf{R}$ matrix is expressed as

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{3.23}$$

Equation (3.22) is now an underconstrained system of 9 equations in 12 unknowns. It is known, however, that the unknowns in the rotation matrix are not independent. The following constraints exist (as shown by Ganapathy [21]):

$$r_{11}^2 + r_{12}^2 + r_{13}^2 = r_{21}^2 + r_{22}^2 + r_{23}^2 = r_{31}^2 + r_{32}^2 + r_{33}^2 = 1, \tag{3.24}$$

$$
\begin{aligned}
r_{13} &= r_{21}r_{32} - r_{22}r_{31}, \\
r_{23} &= r_{12}r_{31} - r_{11}r_{32}, \\
r_{33} &= r_{11}r_{22} - r_{12}r_{21}.
\end{aligned} \tag{3.25}
$$

Note that since the three points are coplanar in the camera coordinate system we can assume $z'_i = 0,\ i = 1, 2, 3$, allowing us to use the image coordinates for points $\mathbf{p}'_i$. Equation (3.22) can now be written as

$$
\begin{aligned}
x_i &= r_{11}x'_i + r_{12}y'_i + t_x \\
y_i &= r_{21}x'_i + r_{22}y'_i + t_y \\
z_i &= r_{31}x'_i + r_{32}y'_i + t_z
\end{aligned} \tag{3.26}
$$

and in matrix form as

$$
\mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.27}
$$

with

$$
\begin{aligned}
\mathbf{A} &= \begin{bmatrix}
x'_1 & y'_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & x'_1 & y'_1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & x'_1 & y'_1 & 0 & 0 & 1 \\
x'_2 & y'_2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & x'_2 & y'_2 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & x'_2 & y'_2 & 0 & 0 & 1 \\
x'_3 & y'_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & x'_3 & y'_3 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & x'_3 & y'_3 & 0 & 0 & 1
\end{bmatrix}, \\
\mathbf{x} &= (r_{11}, r_{12}, t_x, r_{21}, r_{22}, t_y, r_{31}, r_{32}, t_z)^T, \\
\mathbf{b} &= (x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3)^T.
\end{aligned} \tag{3.28}
$$

Provided that the three points are not collinear the matrix $\mathbf{A}$ will not be singular and $\mathbf{x}$ can be solved for a unique solution. After solving for $\mathbf{x}$, equations (3.25) can be used to solve for r_{13}, r_{23} and r_{33}.

3.3.2 Relative Calibration

As an alternative to calibrating each camera relative to the real-world coordinate system it is possible to calibrate the cameras relative to one another. To achieve this calibration the fundamental matrix is first introduced, followed by how it may be used to calculate the camera matrices and finally how it may be calculated.

3.3.2.1 Fundamental Matrix

To perform this calibration the 3×3 fundamental matrix, $\mathbf{F}$, between two cameras C_1 and C_2 is useful. The defining condition for $\mathbf{F}$ is that given any pair of corresponding points in two images, $\mathbf{x} \leftrightarrow \mathbf{x}'$, the following holds true:

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0. \tag{3.29}$$

The fundamental matrix captures the relationship between points on the two image planes resulting from the two cameras. A variation of the fundamental matrix is the essential matrix, $\mathbf{E}$, which relates only the geometric relationship between the two cameras, as opposed to the fundamental matrix that also includes information on the internal parameters. The defining condition for the essential matrix is

$$\mathbf{y}'^T \mathbf{E} \mathbf{y} = 0, \tag{3.30}$$

with $\mathbf{y}$ and $\mathbf{y}'$ the normalized coordinates of $\mathbf{x}$ and $\mathbf{x}'$ respectively, i.e. $\mathbf{y} = \mathbf{K}^{-1}\mathbf{x}$ and $\mathbf{y}' = \mathbf{K}'^{-1}\mathbf{x}'$. A simple relation between $\mathbf{F}$ and $\mathbf{E}$ exists:

$$\mathbf{E} = \mathbf{K}'^T \mathbf{F} \mathbf{K}. \tag{3.31}$$

For all essential matrices the first two singular values are equal and the third is zero. It can be written in the form $[\mathbf{t}]_\times \mathbf{R}$ as shown in [26] ($[\mathbf{t}]_\times$ is the skew-symmetric cross-product matrix of the translation vector $\mathbf{t}$, and $\mathbf{r}$ is the rotation matrix relating the two camera coordinate systems). Having defined the fundamental and essential matrices the next section explains how they can be used.

3.3.2.2 Extracting the Camera Matrices

When using the fundamental matrix the camera matrices can be recovered only up to a projective ambiguity. The essential matrix, however, is able to recover the matrices up to scale [26, p. 256].

Without loss of generality it may be assumed that the first camera matrix is $\mathbf{P} = [\mathbf{I} \mid \mathbf{0}]$ and the second camera matrix is $\mathbf{P}' = [\mathbf{R} \mid \mathbf{t}]$ to make calculation of the second matrix $\mathbf{P}'$ simpler. From section 3.3.2.1 it is clear that the SVD of $\mathbf{E}$ is $\mathbf{U}\ \mathrm{diag}(1, 1, 0)\ \mathbf{V}^T$. This allows for two factorizations of the form $\mathbf{E} = \mathbf{SR}$:

$$\mathbf{S} = \mathbf{U}\mathbf{Z}\mathbf{U}^T,\ \mathbf{R} = \mathbf{U}\mathbf{W}\mathbf{V}^T \text{ or } \mathbf{U}\mathbf{W}^T\mathbf{V}^T \tag{3.32}$$

with

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.33}$$

From this factorization the translation, $\mathbf{t}$, can be determined from the skew-symmetric matrix$\mathbf{S} = [\mathbf{t}]_\times$. Since $\mathbf{St} = \mathbf{0}$ it follows that $\mathbf{t} = \mathbf{U}(0, 0, 1)^T = \mathbf{u}_3$, which is the last column of $\mathbf{U}$. The sign of $\mathbf{E}$, and consequently of $\mathbf{t}$, cannot be determined giving $\mathbf{t} = \pm\mathbf{u}_3$. This result along with the factorization of $\mathbf{E}$ given in (3.32) gives four possibilities for $\mathbf{P}'$:

$$\mathbf{P}' = [\mathbf{U}\mathbf{W}\mathbf{V}^T | +\mathbf{u}_3] \text{ or } [\mathbf{U}\mathbf{W}\mathbf{V}^T | -\mathbf{u}_3] \text{ or } [\mathbf{U}\mathbf{W}^T\mathbf{V}^T | +\mathbf{u}_3] \text{ or } [\mathbf{U}\mathbf{W}^T\mathbf{V}^T | -\mathbf{u}_3]. \tag{3.34}$$

The four solutions correspond to four possible geometric configurations for the two cameras. The difference between terms with $+\mathbf{u}_3$ and $-\mathbf{u}_3$ is simply a reversal in the direction of the vector between to two cameras. The relation between terms with $\mathbf{W}$ and $\mathbf{W}^T$ amounts to a 180° rotation about the line joining the two camera centers.

It is possible to check which of the four solutions is correct by reconstructing a single point using the four options. In only one of the reconstructions will the point be in front of both cameras which gives the correct solution. Figure 3.5 illustrates this.

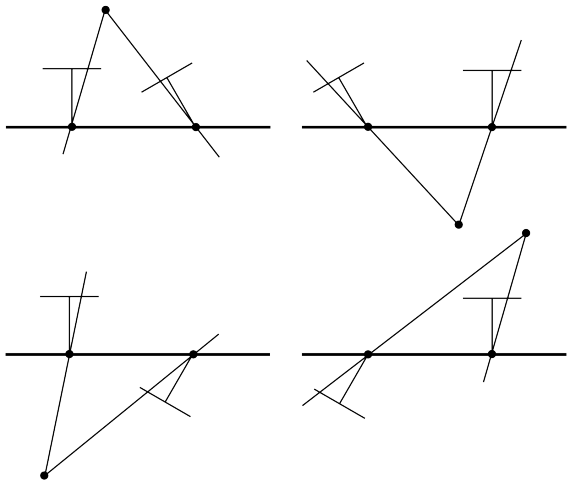


Figure 3.5: Reconstructing a single point from four possible solutions to $\mathbf{P}'$.

3.3.2.3 Calculating the Fundamental Matrix

Given several point matches $\mathbf{p}_i \leftrightarrow \mathbf{p}'_i$ the basic fundamental matrix equation $\mathbf{p}'^T\mathbf{F}\mathbf{p} = 0$ can be used to solve for $\mathbf{F}$. Writing $\mathbf{p} = (x, y, 1)^T$ and $\mathbf{p}' = (x', y', 1)^T$ each point correspondence gives one linear equation in the elements of $\mathbf{F}$:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0, \tag{3.35}$$

where f_{ij} is the element of $\mathbf{F}$ in row i and column j. With a set of n point matches (3.35) can be expressed as a matrix equation:

$$\mathbf{A}\mathbf{f} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = 0. \tag{3.36}$$

The vector $\mathbf{f}$ can now be solved using standard linear algebra techniques (such as an SVD).

3.4 Triangulation

When there are several cameras capturing the same scene it becomes possible to obtain 3D data about objects in the scene. The process of determining the 3D coordinates of an object that is visible

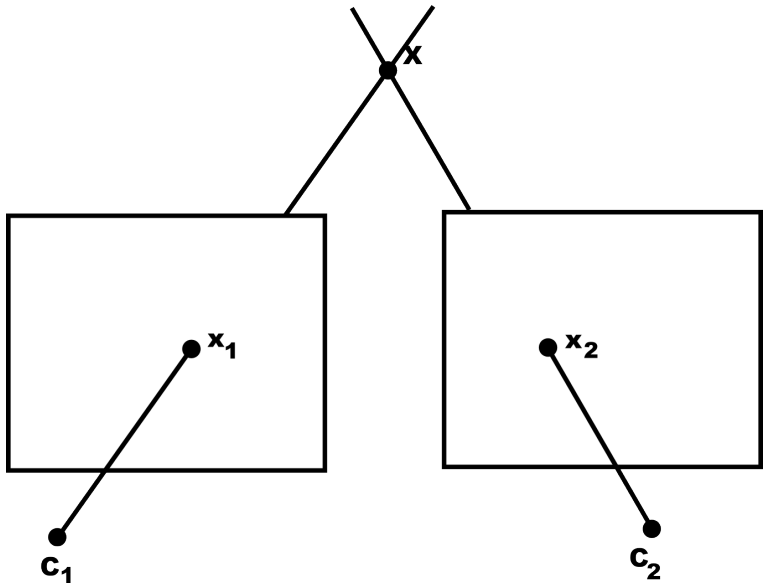


Figure 3.6: Triangulation using point correspondences. The triangulated point is the intersection of the projected lines through the image points.

in two or more views is known as triangulation. The rest of this section discusses how triangulation can be done using firstly two cameras and then n cameras.

3.4.1 Two View Triangulation

Every point, $\mathbf{x}$, on an image plane corresponds to a line, $\mathbf{l}$, in 3D space. Any point on this $\mathbf{l}$ will project onto $\mathbf{x}$. If one is able to find corresponding points in two or more camera views, this information can be used to find the point in 3D space that corresponds to all of the image points. This point is simply the point of intersection between all of the projected lines of each camera. Figure 3.6 illustrates this.

To solve for the point $\mathbf{X}$ we first notice that, given two camera matrices $\mathbf{P}$ and $\mathbf{P}'$ and corresponding points $\mathbf{x} = (x, y, z)^T$ and $\mathbf{x}' = (x', y', z')^T$ in the images:

$$\mathbf{x} = \mathbf{P}\mathbf{X} \quad \text{and} \quad \mathbf{x}' = \mathbf{P}'\mathbf{X}. \tag{3.37}$$

These equations cannot be solved directly as we are working in homogeneous coordinates (the equality sign merely indicates that vectors on the two sides of the equation point in the same direction). To solve the system we take the cross product of the two sides of the equation, i.e.

$\mathbf{x} \times \mathbf{PX} = \mathbf{0}$. From this we get three equations:

$$y\mathbf{p}_3^T\mathbf{X} - z\mathbf{p}_2^T\mathbf{X} = 0, \tag{3.38}$$

$$z\mathbf{p}_1^T\mathbf{X} - x\mathbf{p}_3^T\mathbf{X} = 0, \tag{3.39}$$

$$x\mathbf{p}_2^T\mathbf{X} - y\mathbf{p}_1^T\mathbf{X} = 0. \tag{3.40}$$

Using only the first two equations of each cross product(the third is a linear combination of the first two) we can write it as follows:

$$\mathbf{AX} = \mathbf{0}, \tag{3.41}$$

where

$$\mathbf{A} = \begin{bmatrix} y\mathbf{p}_3^T - z\mathbf{p}_2^T \\ z\mathbf{p}_1^T - x\mathbf{p}_3^T \\ y'\mathbf{p}'^T_3 - z'\mathbf{p}'^T_2 \\ z'\mathbf{p}'^T_1 - x'\mathbf{p}'^T_3 \end{bmatrix}.$$

Since we are measuring directly from the image plane in Euclidean coordinates we have $z = z' = 1$. Using SVD we can solve for $\mathbf{X}$ by finding the right singular vector that is associated with the smallest singular value. This simply corresponds to the right-most column of $\mathbf{V}$ in the SVD.

3.4.2 Optimizing Triangulation Results

In general two lines in 3D space do not necessarily intersect at some point. This is also true for the projection lines used in the triangulation method described in section 3.4.1. Problems may arise from errors in calibration, as well as non-perfect point detection. This causes a situation as shown in figure 3.7 where the projected lines do not intersect each other.

The challenge now becomes to find some best fit solution in the presence of these two factors. When optimizing a system one tries to minimize some cost or error function. For triangulation we have two distances (error values) that can be minimized.

The first is the total projection error, e_p, of the triangulated point $\mathbf{x}$ on each of the projection lines $\mathbf{l}_i$. The projection of a point $\mathbf{x}$ on a line $\mathbf{l}$ where $\mathbf{l} = \mathbf{p} + k\mathbf{n}$ ($\mathbf{p}$ is a point on the line and $\mathbf{n}$ a

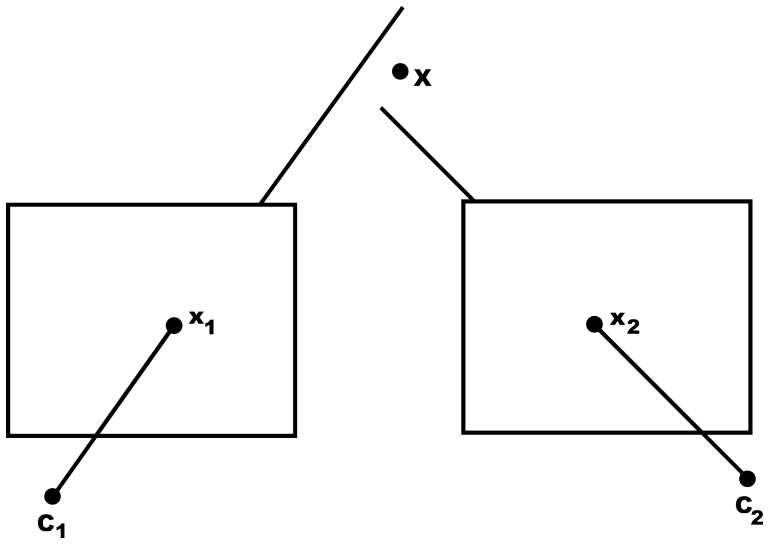


Figure 3.7: Errors in calibration and point detection may result in projection lines that do not cross in 3D space.

unit vector in the direction of the line) is $\mathbf{y}_i = \mathbf{n}_i\mathbf{n}_i^T(\mathbf{x}_i - \mathbf{p}_i) + \mathbf{p}_i$. The total error now becomes

$$e_p = \sum_i \|\mathbf{y}_i - \mathbf{x}\|^2. \tag{3.42}$$

This error value is calculated in the projected space, and as such it is not projectively invariant. The lack of projective invariance implies that the measurement made does not have a concrete physical meaning. An error measurement of 5 may be 5 centimeters in one case and 5 kilometers in another.

The second error function commonly used is the back projection error e_{bp}. This is obtained by calculating the total error by projecting the triangulated point $\mathbf{X}$ back onto each image plane and measuring the distance between the original point, $\mathbf{p}_i$, and the back projected point, $\mathbf{p}'_i$:

$$e_{bp} = \sum_i \|\mathbf{p}_i - \mathbf{p}'_i\|^2. \tag{3.43}$$

This error is measured directly on the image plane in Euclidean coordinates, giving us a projectively invariant distance measure.

In two views it is commonly preferred to minimize the back projection error when optimizing triangulation results. One approach to minimizing the back projection error for two views is known as Sampson correction and is described in [26]. This approach has been extended to three views by Byrod et al. [7] using the Grobner basis [1]. There is, however, no such technique for minimizing the back projection error in more than three views. In the next section we describe a method for minimizing the projection error in multiple views.

3.4.3 Multiple View Triangulation

Triangulation from multiple views presents new challenges, but also some benefits above two view triangulation. On the one hand multiple views provide more information, allowing for more accurate triangulation. On the other hand it is harder to combine the data in a computationally inexpensive manner while keeping a high degree of accuracy. As mentioned in section 3.4.2 we try to minimize the projection error e_p from (3.42).

The first step in minimizing e_p is to find the projection of the point $\mathbf{X}$ on the projection lines, $\mathbf{l}_i$, from each camera. Each of the lines $\mathbf{l}_i$ can be written as $\mathbf{l}_i = \mathbf{p}_i + k\mathbf{n}_i$ where $\mathbf{p}_i$ is a point on the line and $\mathbf{n}_i$ is a unit vector in the direction of the line. To find this representation one begins with the camera equation:

$$\mathbf{x} = \mathbf{KR}[\mathbf{I}|-\tilde{\mathbf{C}}]\mathbf{X} = \mathbf{KR}\tilde{\mathbf{X}} - \mathbf{KR}\tilde{\mathbf{C}} \tag{3.44}$$

which can be rewritten as

$$\tilde{\mathbf{X}} = \mathbf{R}^T\mathbf{K}^{-1}\mathbf{x} + \tilde{\mathbf{C}}. \tag{3.45}$$

The camera center, $(0,0,0)^T$, and the point $(x,y,1)$ on the image plane both lie on the projection line. By substituting them for $\mathbf{x}$ in equation (3.45), two points, $\mathbf{q}_1$ and $\mathbf{q}_2$, are found in the real-world coordinate system that both lie on this line. It is now possible to solve for $\mathbf{p}_i$ and $\mathbf{n}_i$:

$$\mathbf{p}_i = \mathbf{q}_1 \tag{3.46}$$

$$\mathbf{n}_i = \frac{\mathbf{q}_1 - \mathbf{q}_2}{||\mathbf{q}_1 - \mathbf{q}_2||} \tag{3.47}$$

The projection, $\mathbf{y}_i$, of $\mathbf{X}$ on $\mathbf{l}_i$ can then be found as

$$\mathbf{y}_i = \mathbf{n}_i\mathbf{n}_i^T(\mathbf{X} - \mathbf{p}_i) + \mathbf{p}_i, \tag{3.48}$$

with the total projection error

$$e_p = \sum_i \|\mathbf{y}_i - \mathbf{X}\|^2. \tag{3.49}$$

We now want to find $\mathbf{X}$ that minimizes e_p:

$$\begin{aligned} e_p &= \sum_i \|\mathbf{n}_i\mathbf{n}_i^T(\mathbf{X}-\mathbf{p}_i)+\mathbf{p}_i-\mathbf{X}\|^2 \\ &= \sum_i \|(\mathbf{n}_i\mathbf{n}_i^T-\mathbf{I})\mathbf{X}-(\mathbf{n}_i\mathbf{n}_i^T-\mathbf{I})\mathbf{p}_i\|^2 \\ &= \sum_i \|\mathbf{A}_i\mathbf{X}-\mathbf{b}_i\|^2 \\ &= \sum_i (\mathbf{A}_i\mathbf{X}-\mathbf{b}_i)^T(\mathbf{A}_i\mathbf{X}-\mathbf{b}_i) \\ &= \sum_i \left[(\mathbf{A}_i\mathbf{X})^T(\mathbf{A}_i\mathbf{X})-2(\mathbf{A}_i\mathbf{X})^T\mathbf{b}_i+\mathbf{b}_i^T\mathbf{b}_i\right]. \end{aligned} \tag{3.50}$$

Taking the derivative of (3.50) with respect to $\mathbf{X}$ and setting it equal to zero yields

$$\begin{aligned} \frac{\partial e_p}{\partial \mathbf{X}} = \sum_i \left[2(\mathbf{A}_i^T\mathbf{A}_i)-2\mathbf{A}_i^T\mathbf{b}_i\right] &= 0 \\ \sum_i (\mathbf{A}_i^T\mathbf{A}_i)\mathbf{X} &= \sum_i \mathbf{A}_i^T\mathbf{b}_i \\ \left(\sum_i \mathbf{A}_i^T\mathbf{A}_i\right)\mathbf{X} &= \sum_i \mathbf{A}_i^T\mathbf{b}_i \\ \mathbf{C}\mathbf{X} &= \mathbf{d}. \end{aligned} \tag{3.51}$$

Equation (3.51) is now in a familiar form, allowing us to solve it using standard linear algebra techniques.

This chapter looked at the model used to describe a camera and how to use that model to perform some calculations needed to locate players on the field. In the next chapter methods for detecting people and objects on the field are discussed.

Chapter 4

Object Detection

Detecting and locating people in an image or video sequence has many real-world applications, over a wide range of fields. Security systems attempt to detect people in order to alert personnel or automatically begin recordings. Large building complexes such as shopping centers and airports use systems to detect people to gather statistics about traffic flow and queue lengths in an attempt to improve building layout. Even some hand-held cameras are able to detect people, allowing for example the auto-focus to focus on them.

With such a wide range of applications much work has been done to refine the algorithms that are used. The ever present tradeoff between speed and accuracy also plays an important role in algorithm implementation, where an advanced security system with high processing power might use a high accuracy, computationally expensive algorithm whilst hand-held cameras opt for inexpensive algorithms.

Two broad fields for detecting people or objects in video sequences are discussed in this chapter. The first is to find people by trying to match some area in the image to a model of what a person looks like, known as detection by recognition. A second method tries to find areas of motion in an otherwise static scene. In general some further processing would need to be done to confirm that motion is caused by a person, however for the focus of this study the only movement on a sports field will be caused by the players, ball and referees on the field.

4.1 Detection by Recognition

Detection by recognition attempts to recognize a person or object by matching an area of the scene to some model of what a person looks like. Different algorithms use many different features, some attempting to interpret the scene much as a person would, while others analyze the scene using mathematical models.

4.1.1 Edge Orientation Histogram

A popular approach used when detecting specific objects in cluttered scenes is to build some mathematical model that describes the object in a manner that it can then be searched for in an image. One such approach is to model an object (or person) as a collection of edges in different directions, as in [18]. When trying to detect if a person occurs in the scene one then only needs to search for an area in the scene that has a similar collection of edges.

The first step in this method is to build the edge model of a person from a set of training data. Each of the training images will contain a person on a plain background so that the only edges in the image belong to the person being modelled. Edge detection is then performed on the image to reduce it to an edge image. A popular edge detection method is to convolve an image with the Sobel operators [17]:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{4.1}$$

Two edge images, E_x and E_y, are composed, one for each of the Sobel operators. For each pixel in the original image a gradient magnitude and direction can be calculated as:

$$M(x,y) = \sqrt{E_x(x,y)^2 + E_y(x,y)^2}, \tag{4.2}$$

$$G(x,y) = \arctan \frac{E_y(x,y)}{E_x(x,y)} \times \frac{180}{\pi}. \tag{4.3}$$

A histogram can now be constructed by thresholding the edge magnitudes and then collecting the remaining edges into a histogram where the bins represent the edge directions. It may be useful to normalize the histogram to avoid problems that may arise from varying scale. By repeating

this process for each image in the training sequence the model can then be chosen as the average histogram over the sequence.

After the model has been created new images can be processed by first decomposing the image into edge images and computing the edge magnitude and direction for each pixel. After this has been done the image can be searched through for an area that produces a histogram similar to that of the model.

Edge orientation histograms have been used and expanded upon in a number of ways. A notable modification to the standard edge histogram was done by Dalal et al. [13]. In their work they build grids of Histograms of Oriented Gradients (HOG) to detect people in a scene. A search area is broken into several smaller overlapping blocks and edge histograms are calculated for each block. Combining the blocks and normalizing over the entire cell provides more accurate detection results than using only edge orientation histograms.

One severe limitation of using edge orientation histograms is that a histogram may change drastically as the pose of a person changes. This limits the use of edge histograms to situations where the pose of a person does not vary greatly, e.g. in pedestrian detection.

4.1.2 Scale Invariant Feature Transform

Lowe [35] has proposed a method of object recognition by classifying an object as a collection of interest points in a geometric relation, known as the Scale Invariant Feature Transform (SIFT). The object is then found in an image by searching for a collection of similar interest points in a similar configuration.

To locate interest points a difference of Gaussian function is applied to the image at various sampling levels. At the first sampling level the image is smoothed using two passes of a 1D Gaussian function:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-x^2/(2\sigma^2)) \tag{4.4}$$

in the vertical and horizontal directions.

The initial image is convolved with the Gaussian function to give a smoothed image A. This

image A is again smoothed to produce image B. The difference of Gaussian is then taken as $A - B$ (pixelwise subtraction). The next sampling level is generated by sampling the smoothed image B using bilinear interpolation with a pixel spacing of 1.5, and each subsequent level generated by sampling the previous level in the same manner.

Maxima and minima of the scale space are then determined by comparing each pixel with its eight neighbours in its level. If it is a maximum (or minimum) in its level it is compared to the pixels in the level above and below it. If the closest match to the pixel in both those levels is also a maximum (or minimum) it is said to be a scale space maximum (or minimum) and thus an interest point.

At each of the interest points a SIFT key is generated by calculating the pixel edge magnitude, $M(x, y)$, and orientation, $R(x, y)$, of the base image A as:

$$M(x,y) = \sqrt{(A(x,y) - A(x+1,y))^2 + (A(x,y) - A(x,y+1))^2}, \quad (4.5)$$

$$R(x,y) = \arctan \frac{A(x,y) - A(x+1,y)}{A(x,y+1) - A(x,y)}. \quad (4.6)$$

Objects can now be detected in cluttered images by searching for a collection of similar keys in a similar configuration to that of the model set. To make this search easier and to allow for rotation invariance the keys are stored using the radial coordinate system.

The SIFT features have proven to be highly accurate, but suffer from high computational complexity. The features are also only able to handle object rotation up to 20 degrees and scaling up to about 20 percent.

4.1.3 Supervised Learning

Supervised learning has been used extensively in computer vision problems, with object detection being no exception (see [49] for an overview of supervised learning methods). One of the advantages of supervised learning techniques is that the system may be trained on both object and non-object instances. Training with non-object classes may allow the system to reject false positives with greater accuracy.

A supervised learning approach is used by Papageorgiou et al. [41] to detect people, faces and

cars in cluttered scenes. Objects are represented by an overcomplete dictionary of local, oriented, multiscale intensity differences between neighbouring regions. The dictionary is built using the Haar wavelet transform [24], which is a simple wavelet transform based on the Haar wavelet functions.

The learning approach derives a model of the object by training a support vector machine (SVM) using both positive and negative examples. SVMs are a set of supervised learning methods that analyze data and recognize patterns. Given a set of training examples that are classed as either object or non-object the SVM algorithm builds a model that will predict whether a new example falls into the object or non-object categories. For details on SVMs see [49].

Supervised learning produces good results when trained well. Desired objects can be modelled accurately as the algorithm is given many samples to learn from. By training with non-object instances as well, the system is also able to correctly exclude incorrect objects that may be similar to the desired object. Supervised learning has problems, however, in that the system is only able to detect object instances that it has been trained on. Sports players may assume a large number of poses and may play in widely varying environments (day or night, in rainy or cloudy weather, etc). This requires that the algorithm be trained for each pose in each situation, which may be impractical for real-world application.

4.1.4 Parts Detector

Another approach to detecting people in images is to view a person as a collection of flexible parts. Detecting a person in the scene then becomes a case of detecting individual parts and checking if they appear in an appropriate geometric configuration.

One such approach is followed by Mikolajczyk et al. [37] where detection is done using a probabilistic assembly of parts. Seven body parts are used for classification, with each part described by a set of SIFT features. The model for each part is trained using the AdaBoost algorithm [19].

When detecting people in a scene, individual body parts are first searched for. When parts have been found the part with the highest likelihood score is chosen as a starting point. A neighbourhood is searched for other parts that would make up the rest of the body. If all the parts are found a probabilistic decision is made using a joint likelihood model based on two observations: detected

parts and the relative position of the parts.

A Bayesian decision for a body B with features F and geometric parameters R is then made:

$$\frac{p(B|R,F)}{p(\neg B|R,F)} = \frac{p(R|F,B)}{p(R|F,\neg B)} \frac{p(F|B)}{p(F|\neg B)} \frac{p(B)}{p(\neg B)}. \tag{4.7}$$

One of the advantages of such a parts detector is that the entire body need not be visible. By detecting a subset of the total body parts the system may be able to detect a partialy occluded body. Drawbacks to the system exist though. The parts detector is computationally expensive, hindering real-time applications. The parts detector may also fail when players occlude one another as multiple body parts of each person are detected but the system is unable to match parts to specific bodies.

4.2 Motion Detection

Motion detection is the process of detecting and locating areas of motion in a video sequence. Basic motion detection algorithms might only detect if there is any motion in the video sequence, while more advanced algorithms try to find where in each frame the motion occurs.

When the goal is to simply detect motion in a sequence without locating the motion it is possible to view the image as a whole and perform computations on it. Some solutions include comparing eigenvalues and -vectors or singular values of the pixel intensities between frames. Large changes in these values between frames can indicate that there is a change in the scene implying that motion has occurred.

More advanced algorithms try to find where in each frame motion occurs. These algorithms typically attempt to split each frame into two areas: foreground and background. Foreground regions are those areas where motion occurs while background regions are static areas. In order to accomplish this a more detailed comparison of pixels and/or regions of each frame needs to be made. Some popular motion detection algorithms are now discussed.

4.2.1 Pixel Subtraction

Pixel subtraction is a basic form of motion detection. A model of the background is made by taking some weighted average of the pixel intensity for each pixel in the t^{th} frame over some set of previous frames. Motion can now be found by comparing each pixel, $I_t(i,j)$, to the corresponding model image pixel $B_t(i,j)$. A motion image, M_t, can be composed by combining all the results from each pixel comparison:

$$M_t(i,j) = |I_t(i,j) - B_t(i,j)|. \tag{4.8}$$

There are a number of ways in which the background image can be created. Several frames of an empty scene can be used to build up the model which will then remain constant for the duration of the sequence. Each of the initial frames will then be given equal weight:

$$B(i,j) = \frac{1}{n}\sum_{k=1}^{n} I_k(i,j). \tag{4.9}$$

Alternatively the background model may be updated as the sequence continues. This is useful for incorporating permanent changes in the background. It is also useful for situations where a sequence of an empty scene is not available. For this one can use the average of the past n frames:

$$B_t(i,j) = \frac{1}{n}\sum_{k=t-n+1}^{t} I_k(i,j). \tag{4.10}$$

The weighting on each frame can also be used to implement a fading memory where more emphasis is placed on newer frames and older frames are gradually reduced in importance:

$$B_t(i,j) = \sum_{k=t-n+1}^{t} w_k I_k(i,j), \tag{4.11}$$

where $w_n > w_m$ if $n > m$ and $\sum w_k = 1$. This method uses a fading memory on the past n frames. To use a fading memory on all the past frames the following background model can be used:

$$B_t(i,j) = wI_t(i,j) + (1-w)B_{t-1}(i,j), \tag{4.12}$$

with $0 < w < 1$.

The motion image will typically be thresholded and any pixels that fall above the threshold classified as movement.

Pixel subtraction is a basic algorithm, and as such it is computationally inexpensive. This is desirable as it will reduce processing time, increasing the achievable framerate of the system. The basic algorithm, however, suffers from inaccuracy caused by many incorrect detections, due to its simplicity.

4.2.2 Mixture of Gaussians

Several statistical methods exist as solutions to motion detection problems, where the sequence background is modelled using various techniques. Common methods model the background using a multi-modal probability density function (pdf) for each pixel in a frame [42].

Mixture of Gaussians (MOG) is popular in many machine learning and pattern recognition algorithms as a method to build pdfs. The MOG pdfs are constructed as a linear combination of Gaussian probability functions, allowing for multi-modal behaviour that a single Gaussian would not permit.

When applied to background subtraction problems, the MOG technique attempts to model regions of background by a mixture of K Gaussian distributions (due to an increase in computation complexity, K is usually limited to a value between 3 and 5). Each Gaussian corresponds to an aspect of interest of the pixel, such as intensity or brightness.

Generally a training sequence is required for the MOG technique to provide accurate results. Work as been done on algorithms for initializing a MOG algorithm without prior background information. Examples include Zivkovic [58] and Suter et al. [52]. During the training sequence the K Gaussian distributions are found for each pixel in the image. When the algorithm is run on a real sequence each pixel is compared to the Gaussians. If it is classified as foreground then motion has been found, if not then the corresponding Gaussians will be updated with the new pixel values.

4.2.3 Optical Flow

Optical flow is the process of estimating a 2D motion field between frames in an image sequence, originally developed by Horn et al. [28]. If the field can be calculated accurately motion can be found

by searching for areas where there is a great deal of flow in one direction. Areas with no flow are generally background areas while areas with near constant non-zero flow indicate moving objects. Areas with flow in haphazard directions may also indicate background areas, such as foliage moving in the wind.

Optical flow methods function by attempting to find a match for each pixel p in image I_t with a pixel in image I_{t+1}. Search areas are usually limited to a region surrounding the original pixel in order to reduce computational expense and increase accuracy. If such a match can be found the flow of the pixel is the displacement (d_x, d_y) of the pixel between the two images.

Simple algorithms may look for a perfect or near perfect match between frames and calculate the displacement as the movement in x and y directions:

$$(d_x, d_y) = (I_t(p_x) - I_{t+1}(p_x), (I_t(p_y) - I_{t+1}(p_x)) . \tag{4.13}$$

Another method is to calculate the flow between images by matching a window of size $w \times h$ in image I_t with a window of the same size in image I_{t+1}. The best such match can be found by minimizing the function E:

$$E(d_x, d_y) = \sum_{x=p_x-w}^{p_x+w} \sum_{y=p_y-h}^{p_y+h} (I_t(x, y) - I_{t+1}(x + d_x, y + d_y))^2 . \tag{4.14}$$

More advanced optical flow methods have been developed for different applications with trade-offs made between speed and accuracy for each. A popular optical flow method was developed by Lucas et al. [36] that has been used in tracking applications, whilst Camus [9] designed an optical flow algorithm for real-time processing.

In this chapter various object detection methods have been discussed. This is the first step to tracking players through a video sequence, as they need to be found before they can be tracked. In the following chapter some tracking techniques for following the position of an object in a video sequence are reviewed.

Chapter 5

Tracking

Tracking moving objects through video sequences is of great interest to a wide range of fields. Security applications might want to follow people walking through secure areas or track suspicious packages. Another application is that of traffic flow monitoring, allowing for automatic, intelligent traffic light management.

Two broad categories of tracking exist: tracking-by-detection and filter tracking. In the first case, tracking relies on accurate detection of the object through various frames. The second case uses knowledge about the object's state and state transition equations, to predict where the object is moving to next and make object detection easier and more accurate.

5.1 Tracking-by-Detection

Tracking-by-detection, as the name implies, is the process of tracking an object through various frames by attempting to find the object in each of the frames. If the object has been found in each of the frames the path of the object can be found using the data from each frame.

The object detection can be done using any of the available techniques (refer to chapter 4 for more detail). Using mathematical filters it is possible to improve the results of tracking by detection in both speed and accuracy.

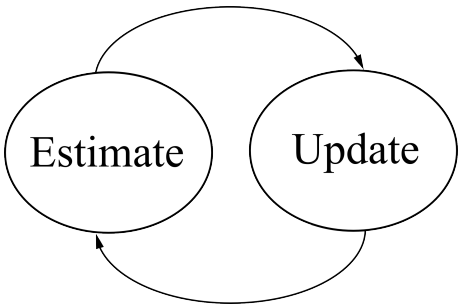


Figure 5.1: Illustration of the Kalman filter process.

One problem with tracking by detection is that detection measurements may suffer from noise distortion. This can cause in the tracked path to appear jagged. Smoothing filters can be used to reduce the impact of detection noise on the path. Some popular filters include the median filter, LULU filters and the Gaussian filters. For more details on mathematical filters see [14].

5.2 Kalman Filter

The Kalman filter [45] estimates a process, or tracks an object, using a form of feedback control. Using knowledge of how the process propagates between stages (or time steps) the filter estimates a process state at some time step k from the filter output at stage $k-1$. This estimate is combined with some noisy measurement at that time to improve the estimate and update the solution. This process is repeated iteratively for as long as required. Figure 5.1 illustrates the Kalman loop.

Because of this estimate-update loop of the Kalman filter, the equations for the filter can be grouped into two categories: estimation and updating. The estimation equations estimate the filter state at a future time step. The update equations then update the estimate after measurements have been made.

First the process and measurement equations must be defined:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mu + \mathbf{w}_{k-1}, \tag{5.1}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{5.2}$$

where $\mathbf{x}_k$ is the process state at time k, $\mathbf{z}_k$ is the measurement at time k and $\mathbf{w}_k$ and $\mathbf{v}_k$ represent

process and measurement noise with normal probability distributions:

$$p(\mathbf{w}) \sim N(\mathbf{0}, \mathbf{Q}), \tag{5.3}$$

$$p(\mathbf{v}) \sim N(\mathbf{0}, \mathbf{R}). \tag{5.4}$$

The matrix $\mathbf{A}$ contains the transition equations between filter states, $\mathbf{B}$ contains the control-input model ($\mathbf{0}$ for purely observational systems) and μ is the control vector. $\mathbf{H}$ is the observation model that maps the true state onto the observed space. The filter equations can now be defined. The estimation equations are:

$$\widehat{\mathbf{x}}_k^- = \mathbf{A}\widehat{\mathbf{x}}_{k-1} + \mathbf{B}\mu_{k-1} \tag{5.5}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}_k, \tag{5.6}$$

and the update equations:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}_k)^{-1} \tag{5.7}$$

$$\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\widehat{\mathbf{x}}_k^-) \tag{5.8}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-. \tag{5.9}$$

In the above equations $\widehat{\mathbf{x}}_k$ is the mean filter state at time k, which serves as the output of the filter. Also note that estimates obtained from the first set of equations are all denoted with a bar-superscript.

$\mathbf{P}_k^-$ and $\mathbf{P}_k$ represent the estimates for the *a priori* (before measurement) and *a posteriori* (after measurement) error covariances. The final filter term, $\mathbf{K}_k$, acts as a weighting factor between the estimated $\widehat{\mathbf{x}}_k^-$ and the measurement. It can be shown that:

$$\lim_{\mathbf{R}\to\mathbf{0}} \mathbf{K}_k = \mathbf{H}^{-1} \tag{5.10}$$

$$\lim_{\mathbf{P}_k^-\to\mathbf{0}} \mathbf{K}_k = \mathbf{0}. \tag{5.11}$$

From this one can see that as the error covariance of the measurement, $\mathbf{R}$, approaches $\mathbf{0}$ (i.e. the measurement becomes more accurate) the weight will favor measured values. In contrast, if the error covariance of the estimate, $\mathbf{P}_k^-$, approaches zero (i.e. the estimate becomes more accurate) the weight will start to favor the estimate.

While the Kalman filter is an optimal filter, it is limited to linear systems with normally distributed noise. This makes it unsuitable in a wide variety of situations. To overcome this limitation the extended Kalman filter (EKF) can be used.

5.3 Extended Kalman Filter

The extended Kalman filter [53] is a non-linear extension of the Kalman filter. It functions in much the same way as the standard Kalman filter by linearizing around the current mean and covariance.

Allowing for non-linear systems, the process and measurement equations become:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mu_{k-1}) + \mathbf{w}_{k-1} \tag{5.12}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k \tag{5.13}$$

with $\mathbf{w}_k$ and $\mathbf{v}_k$ as in (5.3) and (5.4). F and H can now no longer be directly applied to the covariance. Instead at each step a matrix of the partial derivatives, i.e. the Jacobian matrix, is computed and applied. This matrix can then be used in the Kalman filter equations, essentially linearizing the non-linear system. The Jacobians for f and h are:

$$\mathbf{F}_{k-1} = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\widehat{\mathbf{x}}_{k-1}, \mu_{k-1}} \tag{5.14}$$

$$\mathbf{H}_k = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\widehat{\mathbf{x}}_{k-1}^-} \tag{5.15}$$

The estimation equations now become:

$$\widehat{\mathbf{x}}_k^- = f(\widehat{\mathbf{x}}_{k-1}, \mu_{k-1}) \tag{5.16}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{5.17}$$

and the update equations are:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k) \tag{5.18}$$

$$\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\widehat{\mathbf{x}}_k^-)) \tag{5.19}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-. \tag{5.20}$$

While the EKF does address the linearity problems it has several problems of its own. First, it is generally not an optimal filter (except in linear cases where it would be identical to the standard

Kalman filter). A second problem is that a poor initial estimate or process model may cause the filter to diverge quickly due to its linearization. For the EKF to give accurate results the noise present in the system also needs to be normally distributed.

To improve filter performance above that of the EKF the particle filter can be used. The particle filter is able to better approach the Bayesian optimal estimate. It can also be used in the presence of noise that is non-normally distributed.

5.4 Particle Filter

The particle filter [4] is a generalization of the Kalman filter. The filter uses Monte Carlo techniques to solve non-linear filtering problems. The filter is also able to cope with non-Gaussian noise. In this section we provide an overview of the particle filter. For a more detailed discussion, including derivation, the reader is referred to [27].

The particle filter consists of several particles (anywhere from 100 to 1000, depending on the problem). Each of the particles represents a possible solution to the problem, and is represented at time t by some state vector, $\mathbf{p}_{i,t}$. The state vector describes the proposed solution for that particle. At each iteration of the filter, the particles are propagated using some model function with model noise, η_t, added:

$$\mathbf{p}_{i,t} = f(\mathbf{p}_{i,t-1}) + \eta_t. \tag{5.21}$$

where f is a function that describes the movement of the particles between each iteration of the filter.

After all the particles have been moved forward they are compared to some expected solution (or template). This can be done by creating a model of the object that is being tracked as well as for each particle after it has been moved. The particle models can then be compared to the object model (or template). After comparing a particle i to the expected model a dissimilarity measure d_i can be found, such that $d_i = 0$ would indicate an exact match and a greater value of d_i would imply less accuracy.

Every particle is given a weight $w_{i,t}$ for the current frame according to its dissimilarity measure,

as follows:

$$w_{i,t} = exp(-d_{it}^2/(2\sigma^2)). \tag{5.22}$$

The final weight of each particle is then the product of its previous weight and the weight calculated for the current frame. Once all the weights are calculated, they are normalized so that they sum to 1. Note the manner in which the value of σ scales the weights. A large σ means that the values of $d_i^2/(2\sigma^2)$ are closer together. The weights of the particles would then change slowly because the weights of poor matches do not differ greatly from those of good matches. Hence the filter places more emphasis on the model equations than on the particle matches. The opposite holds for a small σ value.

There are two popular choices for the output of the particle filter, which should be the "best" estimate of the object's current position. The first is a best fit solution, where the particle with the highest weight is returned. The second solution is a weighted average of all the particles. The state of each particle is weighted with its w_i value.

An important issue worth mentioning is that of particle degradation. After several iterations the weights of many particles may drop down to zero or values very close to zero as a result of repeated high dissimilarity scores. When a particle weight becomes this low it ceases to contribute to the solution. Even if that particle becomes a very good match at a later stage, its very low weighting prior to that point will prevent its weight from increasing to a meaningful value. To combat this it is useful to perform re-sampling. During re-sampling new particles are chosen from the current set by discarding low weighted particles and splitting high weighted particles into several new particles. All particles are then assigned equal weights and the process continues.

The past several chapters, including this one, have discussed the various components that are needed to design a multi-view 3D tracking system. In the next chapter the components are combined to build a full system for tracking the 3D positions of players on a sports field.

Chapter 6

System Design and Implementation

In the previous chapters various mathematical models and techniques were considered as theoretical background for the problem of tracking players on a sports field using multiple cameras. In this chapter the various components discussed are selected and combined to form a complete system.

First the problem of calibrating multiple cameras observing the same scene is addressed. Next player detection and tracking in a single view is discussed, followed by tracking the players in 3D.

6.1 Calibration Method

As mentioned in section 3.2 there are two options for calibrating several cameras to a single scene: absolute calibration and relative calibration.

If relative calibration is used the cameras will be calibrated against one another, but not the real-world coordinate system. This will allow tracking and triangulation calculations to be made, but all the calculated points will be relative to a camera-coordinate system. To convert these points to the real-world coordinate system a further calibration step will be required. Because of this second calibration step it becomes easier to simply perform absolute calibration on the cameras.

6.1.1 Internal Calibration

The first step is to calibrate the internal camera parameters (those that are included in the **K** matrix). This calibration can be done in an offline environment before the system is deployed to a site. A popular approach to perform this calibration is to use a checker-board pattern [57] where the number and physical size of squares are known.

The corners of each of the squares on the checker-board can either be detected automatically or selected by a human. Calibration can then be performed by using the corners as interest points. As only the internal parameters are calculated the world-coordinate system can be chosen to correspond to the checker-board. See section 3.2 for details on calibrating a camera using $\mathbf{x} \leftrightarrow \mathbf{X}$ point correspondences.

This process can be repeated several times with each iteration providing a set of calibration parameters. An average or least square solution to the set of parameters can then be chosen for each camera.

The discussion here and in chapter 3 assumes the pinhole camera model, in which radial lens distortion is ignored. In a real-world implementation this might be a problem, but can easily be fixed by first dewarping the images prior to further computation. Finding the necessary parameters that enable the dewarping may also be done during internal calibration by a method such as [34].

6.1.2 External Calibration

For external calibration several point correspondences are needed between points with known real-world coordinates and the image coordinates of those points. As mentioned in section 3.2 a minimum of three such point correspondences are required. Any points may be used with the restriction that no three points lie on a single line in space.

A convenient set of points to use is the set of corners made by intersecting lines on the field. These lines can be accurately detected and the corresponding intersections found to give the corners. To detect the lines, and through them the field corners, the Hough transform is used.

The Hough transform [15] is a method to detect lines in an image. The first step is to perform

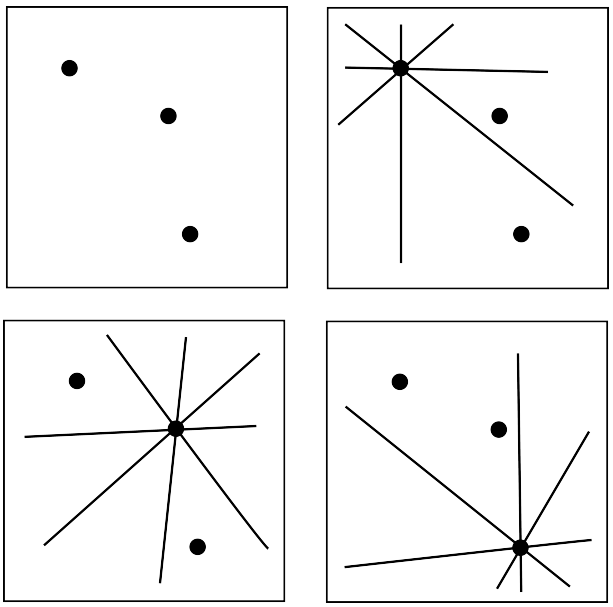


Figure 6.1: Sample lines for building the Hough-space from detected points.

edge detection on the image using any of the available techniques (Sobel [17], Canny [10], etc.). At each edge point, several lines can be drawn as shown in figure 6.1. Each of these lines is then parameterized using two parameters. The first is the distance between the origin, **o**, and the closest point on the line, **c**. The second parameter is the angle between the x-axis and the vector joining **o** and **c**. A two-dimensional histogram can be built where each bin is associated with a distance-angle pairing. Local maxima above some threshold in this histogram will then highlight lines on the field. Figure 6.2 illustrates this process.

Once the lines are found the intersection of those lines can be found by turning to homogeneous coordinates. The standard equation for lines in $\mathbb{R}^2$ is $ax+by+c=0$, allowing each line to be uniquely represented by a coefficient vector $\mathbf{l} = [a, b, c]^T$. The intersection, $\mathbf{p}$, of any two lines represented this way is simply the cross-product between the two lines:

$$\mathbf{p} = \mathbf{l} \times \mathbf{l}'. \tag{6.1}$$

As the real-world coordinates for the corners are known (if the dimensions of the field are known) they make ideal candidates for calibration points using the calibration method of section 3.3.1.

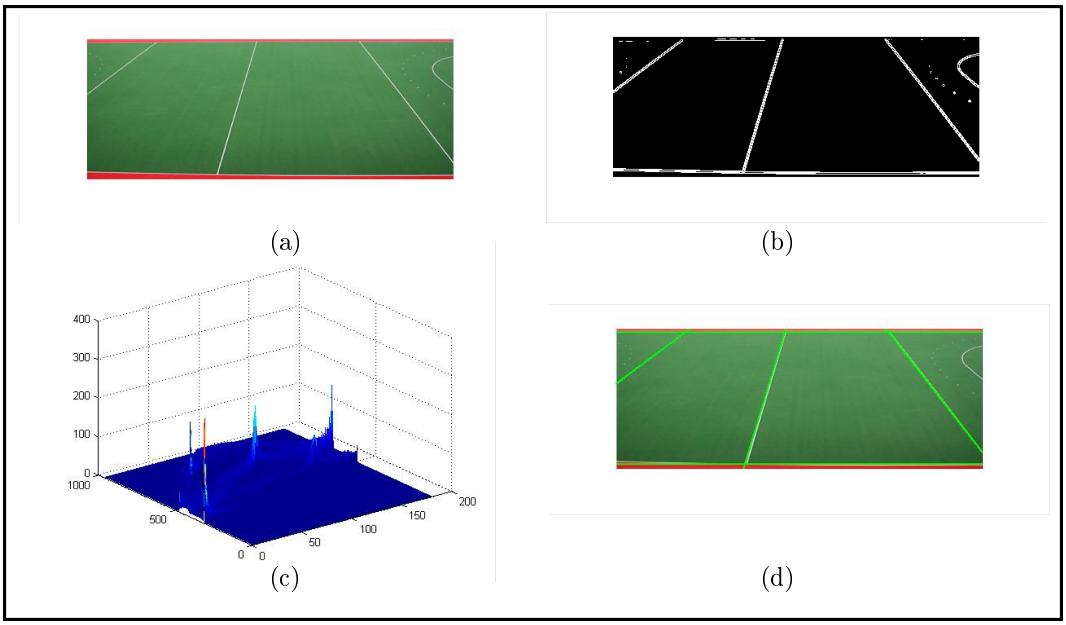


Figure 6.2: Illustration of the Hough transform for line detection: (a) Original Image, (b) Sobel Edge detection, (c) Hough transform histogram and (d) Detected lines.

6.2 Detection and Tracking in 2D

Tracking players in 3D through the use of multiple cameras first requires that the players be tracked in each of the individual camera sequences. To accomplish this players first need to be detected in each camera's field of view separately and then tracked through the video sequence.

A shortend version of this section appears in our paper [51]

6.2.1 Detection

In chapter 4 several techniques for detecting people in images were discussed. Although these methods can achieve high rates of success they are either limited in their application or computationally expensive. On a sports field it is likely that the only moving objects will be players, a ball and possibly a referee. All of these are objects that one may want to track. As such, motion detection is an attractive choice as an effective, low cost detection algorithm.

(a) (b) (c)

Figure 6.3: A depiction of (a) the 2-rectangle feature, and (b) & (c) some 3-rectangle features. An image is convolved with one of these masks. Black indicates regions of negative elements and gray regions of positive elements in the mask.

6.2.1.1 Difference Image

The first step in the motion detection process is background subtraction. A model of the background is created and the current frame is compared to this model (see section 4.2.1). Pixels that differ from this model are interpreted as motion and analyzed further in later stages. One of our main objectives is to have the system run at a high speed, and therefore a simple and fast approach was chosen that turns out to be highly effective.

The background model is constructed using the rectangle features of Viola and Jones [50]. Different rectangle features (vertical, horizontal, 2-rectangle, 3-rectangle, etc.) were tested, and the 2-rectangle feature in figure 6.3 (a) was found to produce excellent results while also being the fastest to calculate.

At each pixel location we calculate the value of the rectangle feature in the region around the pixel. Let $I_s(i,j)$ denote the grayscale intensity of the pixel at row i and column j in frame s of the sequence. Rectangle features can be computed for that frame to give a feature image F_s, where

$$F_s(i,j) = I_s(i,j) + I_s(i+1,j) - I_s(i,j+1) - I_s(i+1,j+1). \tag{6.2}$$

Using these feature images the background model is calculated. The model associated with the current frame k is denoted as B_k. B_k is calculated as the weighted average of the feature images over the past p frames, where p is the memory length of the model,

$$B_k(i,j) = \sum_{s=k-p}^{k-1} w_s \, F_s(i,j). \tag{6.3}$$

The value of p can be varied to fit the expected background situation. As p increases the memory length of the background model also increases. This is useful for static backgrounds as the longer memory length will create a more accurate model. Conversely, if the background often changes a small p can be chosen. This will allow the model to incorporate the changes much faster and update the model as necessary. The symbol w_s in equation (6.3) corresponds to a weight assigned to the frame s. We can assume without loss of generality that $w_{k-p} + w_{k-p+1} + \ldots + w_{k-1} = 1$. The weights can be chosen as constant, causing all frames to have the same influence on the background model. Alternatively one may chose the weights so that $w_{k-p} < w_{k-p+1} < \ldots < w_{k-1}$, which will create a "fading memory" where more recent frames hold greater importance.

Note that the background model is constantly updated, in fact at every frame, to accommodate variations in light intensity, unforeseen camera motion, and motion due to non-player objects such as trees in the background or spectators. The rectangle features are extremely quick to determine, so that this constant updating of the background model is by no means a computational bottleneck.

After specifying a background model B_k for the current frame k, a difference image D_k is created simply as the absolute difference between F_k and B_k, i.e.

$$D_k(i,j) = |F_k(i,j) - B_k(i,j)|. \tag{6.4}$$

Every value in D_k is interpreted as a sort of likelihood of motion occurring at the corresponding pixel. Figure 6.4 shows an example frame and its difference image (normalized for display purposes). Observe that the shadows of the players are hardly visible in the difference image. This is another exceptionally useful effect of the chosen rectangle features. The background model consists of relative pixel differences so that, for example, grass in sunlight and grass under a cast shadow are viewed as being equal (or at least close to being equal).

6.2.1.2 Extracting Areas of Motion

Once the difference image has been created regions of high density are searched for. These high density areas will in general correspond to players that are moving around against the static background.

The first step in finding the high motion density areas is to convolve the difference image with

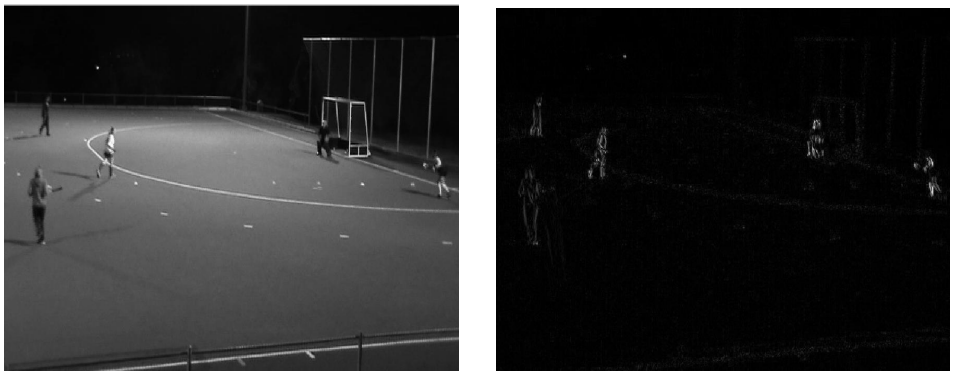
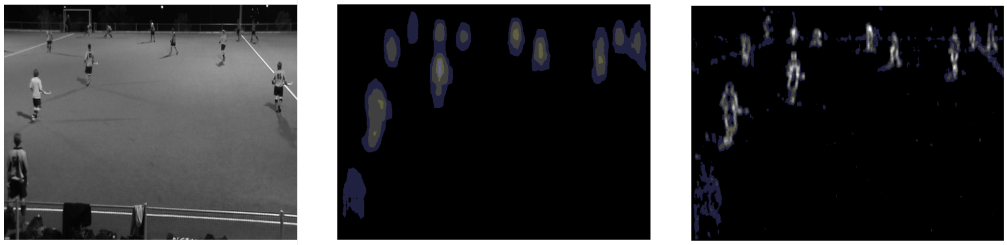


Figure 6.4: One frame from a grayscale video sequence and the difference image resulting from our background subtraction procedure. Note that shadows hardly appear in the difference image.



(a) (b) (c)

Figure 6.5: The effects of changing the averaging filter size. The original motion image (a) is averaged first with a large filter (b) and then a small filter (c). The larger filter only highlights large areas of high motion density.

an averaging filter. The size of the filter, $(2a+1) \times (2b+1)$, can be varied to allow larger or smaller areas of motion to persist. A small filter will highlight small areas of motion as well as large areas of motion, while a large filter will exclude smaller motion patches. Figure 6.5 illustrates this point.

This averaging process serves two purposes. First, the motion values of individual pixels are summed over a neighbourhood so that regions with high motion density stand out. Second, regions containing a small amount of motion, likely due to noise, are de-emphasised.

Figure 6.6 shows on the left the motion image corresponding to the frame in figure 6.4. Regions containing a large amount of motion are clearly visible.

Objects can now be extracted from the motion image by, for example, a blob detection method. A fast and easy technique for isolating different objects, that was found to be very successful, is to simply threshold the motion image by some value t, and then perform a connected-components

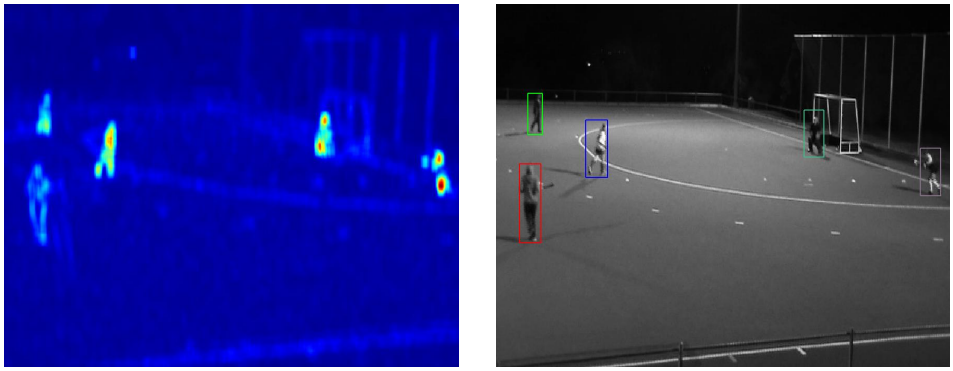


Figure 6.6: A visualization of the motion image corresponding to the frame in the previous figure (left), and objects extracted by the detection of dense motion blobs (right).

labelling. Of course, the choice of t has some effect on the outcome. A smaller threshold may produce many false detections whereas a larger threshold, although less prone to false detections, may miss some of the objects of interest. The effect of t is further explored in section 7.2.1. Figure 6.6 (right) shows a result of this procedure performed on the motion image shown on the left, where every extracted object is highlighted by the bounding box of its corresponding component (or blob) in the motion image.

6.2.1.3 Passing Objects to the Tracker

In the final step of the motion detector, regions of interest as detected in the previous stage are considered. Regions are assigned to one of three categories and are then dealt with accordingly. The three categories are:

1. an object already being tracked;
2. an object not yet tracked but under consideration;
3. an entirely new object.

If an extracted region is in close proximity to a player that is already being tracked, that motion is assumed to correspond to the tracked player and is ignored. The region is then excluded from any further processing at this stage.

Next the "possible object" queue comes under inspection. This queue contains a list of regions that are currently under consideration for tracking. If an object is placed into the queue, and remains in the queue for several frames it can safely be assumed to be an object that needs to be tracked. It is then passed to the tracker for tracking and removed from the possible object queue. Although false motion detections do occur, it is very rare for them to occur in close proximity to each other over several consecutive frames. This queue therefore allows us to avoid most false detections.

Once all regions that fall into category 1 above have been removed, regions of extracted motion are checked to see if they are in close proximity to any regions currently residing in the possible object queue. If this is the case, we increase the frame count for that region in the queue. Any remaining regions (that did not fall into either category 1 or 2) are newly discovered regions. These regions are then added to the possible object queue.

Lastly we look through the possible object queue again. Regions that did not have their frame count updated in the current frame are discarded from the list as they are likely to be false detections from a previous frame. Any remaining regions that have been in the queue for some predetermined time (5 frames or so) are passed to the tracker for tracking.

6.2.2 Tracking

After finding players using the motion detection step, the players need to be tracked through the video sequence. To accomplish this a hierarchical approach to the particle filter is used (as developed in [56]). In this approach objects are represented using several different descriptors. Descriptors vary from coarse, fast descriptors that may yield many false positives, to fine but slow descriptors for more precise classification. Objects are first compared to the coarse descriptors, and if they are a good match are then passed to the slower second stage. This hierarchical approach allows for much faster execution of the filter while maintaining good results.

For this implementation, each particle represents a possible (x, y) location for a player in a frame of the video sequence. Each particle is represented by a four-tuple state vector $\mathbf{p}_{i,t} = (x_{i,t}, y_{i,t}, x'_{i,t}, y'_{i,t})$ where $x_{i,t}$ and $y_{i,t}$ are the current possiotion of the particle and $x'_{i,t}$ and $y'_{i,t}$ are the velocities in the x and y directions. For filter propagation standard movement equations are

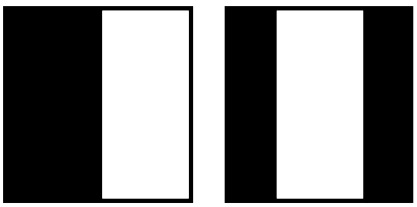


Figure 6.7: Illustration of a rectangle feature used as a tracking descriptor.

used:

$$\mathbf{p}_{i,t} = (x_{i,t-1} + \Delta t x'_{i,t-1},\ y_{i,t-1} + \Delta t y'_{i,t-1},\ x'_{i,t-1},\ y'_{i,t-1}) \tag{6.5}$$

with $x'_{i,t}$ and $y'_{i,t}$ calculated by taking the difference in x and y positions of the filter output after each iteration.

6.2.2.1 First Stage Descriptors

For first-stage descriptors in the hierarchical particle filter the rectangle features of Viola and Jones [50] are once again used, this time those depicted in figure 6.7. The descriptors act similar to a mask that has been run over the image. At each pixel location the intensity values of pixels in the region around the object are added and subtracted. Figure 6.7 and the equation below illustrate how to calculate the feature on the right using a 9×9 block (in our system the block size is closer to 80×40).

$$R(i,j) = \sum_{i=-4}^{4} \left[\sum_{j=-4}^{-3} I(i,j) + \sum_{j=3}^{4} I(i,j) - \sum_{j=-2}^{2} I(i,j) \right]. \tag{6.6}$$

These features are chosen as they are very fast to calculate. Using the integral image [50] further speed increases can be made. The integral image is such that each pixel value is the sum of all the intensity values of all pixels to the left or above the current pixel. This can easily be calculated as follows:

$$s(x,y) = s(x,y-1) + I(x,y), \tag{6.7}$$

$$I_{\int}(x,y) = I_{\int}(x-1,y) + s(x,y), \tag{6.8}$$

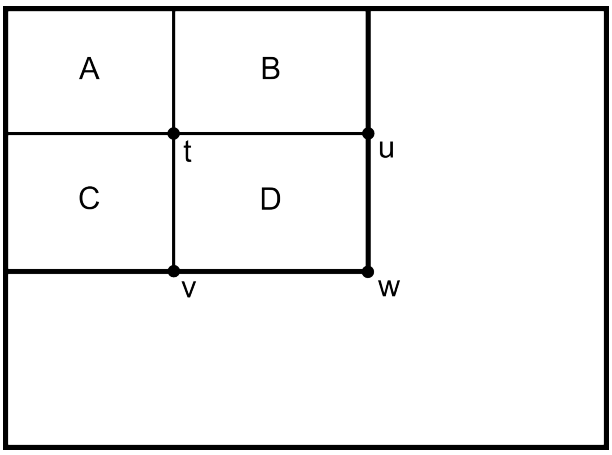


Figure 6.8: Using the integral image to calculate the sum of all the pixels in a rectangle.

where $s(x, y)$ is the sum of pixels in the row thus far ($s(x, -1) = -1$) and $I_{\int}(x, y)$ is the integral image. Using the integral image the sum of any rectangle can be found as the sum of four integral image values. In figure 6.8 the value of the integral image at point t is the sum of the pixels in rectangle A. At point u it is $A + B$, and similarly for points v and w. The sum of the pixel intensities of rectangle D is then $w + t - (u + v)$.

Two rectangle features are calculated for every particle. The dissimilarity measure between the calculated feature and the model feature is taken as the absolute difference between the two. Particle weights for each feature can be calculated using equation (5.22). Each particle now has two first-stage weights assigned to it, one for each feature. The weights can be combined either using the average of the two or by multiplying the two together. An average of the two will give particles that have a good match in both features a high final weight, however particles that have one good and one poor match will still have relatively high scores. Only particles that scored a good match in both features are of interest, making the multiplication method better suited to the problem.

6.2.2.2 Second Stage Descriptors

In the second stage of the hierarchical particle filter only those particles that have a contributable weight after the first stage, i.e. those that have a first-stage weight greater than some threshold, are considered. Particles with small weights after the first stage are considered to be very different to the object, and by ignoring them in the slower second stage we gain computational time. In this stage

a histogram of oriented gradients [13] is calculated as a more precise descriptor than the rectangle features used in the first stage.

The calculation of a histogram of oriented gradients, as in section 4.1.1, requires gradient vectors for each pixel in the image. Discrete derivative operators, such as the Sobel operators, can be used for this purpose and yield two edge images: E_h that highlights horizontal edges and E_v that highlights vertical edges. The magnitude and angle of the gradient vector at each pixel is then calculated as

$$M(i,j) = \sqrt{E_h(i,j)^2 + E_v(i,j)^2}, \tag{6.9}$$

$$G(i,j) = \arctan\left[E_v(i,j)/E_h(i,j)\right]. \tag{6.10}$$

Gradients with a magnitude greater than some threshold are then binned into a histogram according to their angles.

The histograms of all the particles need to be compared with that of the model in order to arrive at some dissimilarity value. There are various ways in which the distance between two histograms can be calculated.

The "city block" and Euclidean distances (i.e. the L_1 and L_2 norms) are fast to compute but do not perform adequately on histograms where the order of the bins carry some meaning. Consider, for example, three histograms $h_1 = [1\ 5\ 1\ 1\ 1\ 1\ 1]$, $h_2 = [3\ 1\ 3\ 1\ 1\ 1\ 1]$ and $h_3 = [1\ 1\ 1\ 1\ 3\ 3\ 1]$. Here h_1 and h_2 should be considered as being much closer to one another than, say, h_1 and h_3. However the Euclidean distance gives $d(h_1, h_2) = d(h_1, h_3) = \sqrt{24}$.

Distances that measure the difference between discrete probability density functions can also be used to compare histograms. Examples include the Kullback-Leibler divergence [33] and the Bhattacharyya distance [5]. These measures, however, also fail for the same reason as the L_1 and L_2 norms.

There are more indicative measures of the distance between histograms. The earth mover's distance (EMD) [43], for example, regards the histograms as piles of dirt and determines the minimum cost required to turn one into the other (where cost is defined as amount of dirt times the distance by which it is moved). This optimization problem, although linear, is rather computationally intensive for the purposes of this problem. Cha and Srihari [11] proposed a measure which is related to the EMD but is much faster to calculate. Because gradient orientations range between 0° and 360°,

with the endpoints regarded as equal, the modulo distance measure (as explained in full detail in [11]) is used.

Particles that were ignored in the second stage due to low first stage weightings still require a second stage weight for the filter to propagate forward. As there is no distance measure calculated for these particles, they are given a second stage dissimilarity value equal to twice the largest value calculated in the second stage. The second stage weights for all the particles can now be calculated using equation (5.22).

6.2.2.3 Filter Output and Updating the Filter Model

Once all the first and second stage weights have been calculated a filter output can be obtained. The first and second stage weights for each particle are multiplied together, after which all the weights are normalized to produce a final weight for each particle. A weighted average of all the particles is taken to find the filter output X:

$$X(x, y) = \sum_{i=0}^{n} w_i p_i(x, y). \tag{6.11}$$

The model that is being tracked must now be updated for the next iteration of the filter. After finding X, the first and second stage descriptors are calculated around that point, and those descriptors are used for the next iteration of the filter.

The next section looks at combining the data from the 2D trackers to estimate the 3D position and track players in real-world coordinates.

6.3 Tracking in 3D

Once players are successfully tracked in 2D it becomes possible to estimate and track their 3D positions. To accomplish 3D tracking the data from the 2D tracking is required for each player in each view. This 3D data can then be fed back to the 2D trackers to check and possibly correct errors in the 2D tracking. This forms a feedback loop as illustrated in figure 6.9.

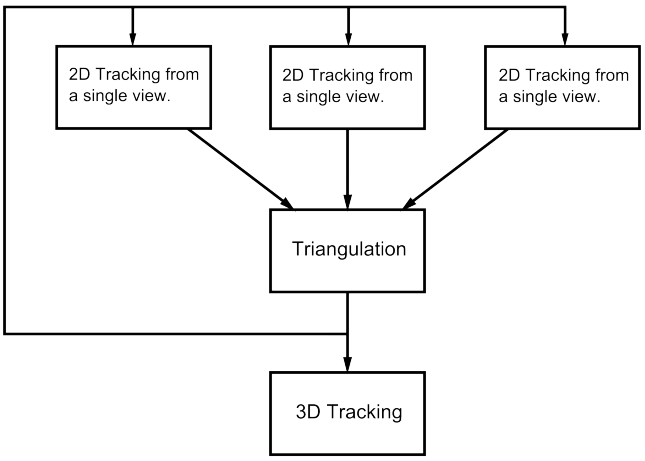


Figure 6.9: Tracking feedback loop: 2D data is used to calculate 3D points which are fed back to the 2D trackers for error correction.

The rest of this section details the processes of combining the various views, triangulating the player positions and the feedback loop.

6.3.1 Matching Players Between Views

Before triangulation of player positions becomes possible it is necessary to match the players between views. Several options exist when trying to accomplish this, such as shape, colour and position.

Shape and colour methods operate by quantifying the shape and/or colour aspects of the person being tracked using, for example, edge or colour histograms. These histograms can be compared to histograms of players being tracked in different views, and should a match be found they are assumed to be the same player in the different views.

These methods can fail, however, when applied to the problem of tracking sports players. Colour methods are ineffective as players on the same team will all be wearing similar clothing. Attempting to match players in this situation will result in multiple matches making it impossible to know which is the correct match. Shape matching on the other hand fails as the player shape may vary drastically when viewed from different angles. Figure 6.10 illustrates the problems that arise when attempting shape or colour matching.

Figure 6.10: Difficulties that arise from matching players between views using shape or colour matching.

Position matching estimates the position of the player on the field from each view individually. This estimation may not be highly accurate, but it does allow one to identify clusters of estimated points. These points will indicate the presence of a player on the field and the corresponding projections of the players in the 2D views.

The single view estimation begins by finding the line in 3D passing through the tracked point on the image plane (see section 3.4.3 for details on calculating the line equations). Once this line has been found the intersection between the line and a plane some distance above the field is calculated. According to [59] the average height of a male in South Africa is 168 cm, indicating that a plane 84 cm above the playing field should be chosen when tracking the center of the player. Similarly the average height of a female in South Africa is 158 cm, indicating a plane 79 cm above the playing field.

After the intersection points have been calculated they can be compared to intersection points from different views. If clusters of intersection points are found close to one another then a match is

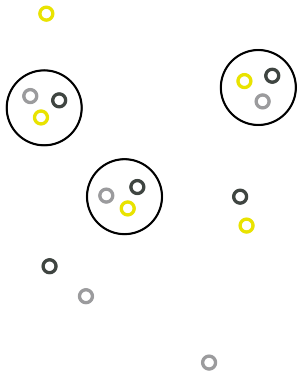


Figure 6.11: Matching players in different views using single view position estimation. Different colours indicate people detected in different views.

made between the different views. See figure 6.11 for an illustration of the process. If two or more intersection points from a single view are located close to each other, i.e. during an occlusion, that location cannot be assigned with a high level of certainty and it is ignored until the two tracked players move away from one another. Also note that once a match has been made, that match remains for the rest of the program execution and the matching step does not need to be repeated.

Another advantage of this matching method is that the calculation of the 3D lines is also required for the triangulation step. This has the effect that the matching step does not greatly increase computational time.

6.3.2 Triangulating Player Positions

Once all the players are tracked in each of the video sequences the positions of players can be triangulated on the field. Two options exist for triangulating from multiple views:

- pairwise, back-projection error minimizing triangulation;
- multi-view, forward-projection error minimizing triangulation.

In the first case the object is triangulated for each possible pair of cameras using back-projection error minimization techniques as discussed in section 3.4.1. This will give $\frac{1}{2}n(n-1)$ solutions when

the object is visible by n cameras. These solutions may then be combined to get a final point by taking the average or least-squares estimate of the set of points. The second option triangulates a single point using all views in a single step, by minimizing the forward projection error rather than the backward projection error.

Tests of the two techniques gave similar results (see section 7.2.4), causing a decision between the two to hinge on computational speed. In this respect the multi-view triangulation is superior to pairwise triangulation due to the fact that multi-view triangulation increases linearly in computational complexity with the number of cameras while pairwise triangulation is of order n^2.

6.3.3 Error Correction

When tracking the 3D positions of players using multiple cameras, this 3D data can be used to increase the accuracy of the tracking in the individual camera scenes. By comparing the 3D position obtained by triangulation to an estimation of the position based only on each view individually it becomes possible to detect and correct errors in the single view tracking.

After triangulating a player based on the 2D tracking results a set of distance measures can be calculated between the triangulated point and the projected point from each camera, using the Euclidean distance. This projected point is the same point as calculated in section 6.3.1 when finding player matches.

If any of the distance measures are greater than some threshold it may indicate that there is a problem with the corresponding 2D tracking. To correct this error the player's location is triangulated a second time, using only tracking results from those trackers where the distance measure is below the threshold. This new 3D point, $\mathbf{X}$, is then projected back to the discredited views using the standard camera equation:

$$\mathbf{x} = \mathbf{P}\mathbf{X}. \tag{6.12}$$

The tracker corresponding to that player in that view can then be restarted at the calculated point $\mathbf{x}$. If less than two of the projection points fall within the threshold then there is no reliable way to determine which of the trackers have failed and which of them are still accurate. In this case the player may need to be dropped from 3D tracking and all corresponding 2D trackers stopped.

The player will then be detected and tracked again as a new entity as if it is a new player on the field.

In this chapter the various components that were discussed earlier were combined to form a complete multi-view 3D tracking system. In the next chapter the individual components, as well as the system as a whole are tested and the results presented.

Chapter 7

Results

This chapter presents the results of the system that has been described in the previous chapters. The individual components that make up the whole of the system are first tested on their own. This allows one to identify weak and strong aspects of the system where further work should focus. After the components have been tested the system as a whole is considered.

7.1 Software

As mentioned in section 1.3, part of the aims for this thesis was to develop the code to implement the overall system. To this end, four modules were developed during the course of the thesis: a calibration module, a detection module, a 2D tracking module and a triangulation module. For the calibration module OpenCV [6] was used to perform the internal calibration while the external calibration was developed manually. Each of the other modules was developed without the aid of external modules. The system could have been developed for a graphical processing unit (GPU) that could push the system into a real-time implementation, but it was considered to be outside the scope of this project.

7.2 Component Results

The component results are broken down into three sections: motion detection, 2D tracking and 3D tracking. Each component is tested to find its various strengths and weaknesses and the results are then discussed.

7.2.1 Motion Detection

The motion detection subsystem was applied to four video sequences of 200 frames each (frames were captured at 640×480, and upsampled to 1280×960 using bicubic interpolation). Sample frames are shown in figure 7.1. Sequences 1 and 2 are real world recordings, whilst sequences 3 and 4 were synthetically generated using the Unreal Development Kit (UDK) [60], recorded with WeGame [61]. The synthetic data is useful for testing purposes since ground truth values are available.

Two quantities were measured: precision and recall. Precision is a measure of how accurate the detection is and is calculated as the number of correct detections (players) divided by the total number of detections. Recall gives an indication of how complete the detection is and gives the number of correct detections divided by the number of objects (in this case players) that are actually present. Of course, ideally these two values should both be as close as possible to 1. In an effort to measure precision and recall, manual annotation of the video frames were performed.

A crucial user-specifiable parameter affecting precision and recall in the motion detection system is the threshold t discussed in section 6.2.1.2. It specifies the amount of motion needed for a blob to be classified as a moving object. Table 7.1 below lists the obtained precision and recall for the four video sequences, for a few different values of the threshold t.

The poor performance of $t = 5$ in all the sequences comes as a result of over-detection. Many regions that do not actually contain moving objects are falsely detected. An increase in t leads to fewer false detections but, at some stage, starts to exclude true objects from being detected (i.e. low recall).

Most of the missed and incorrect detections occur as a result of one player occluding another or when two players are close together. In the first case only the player visible to the camera is detected

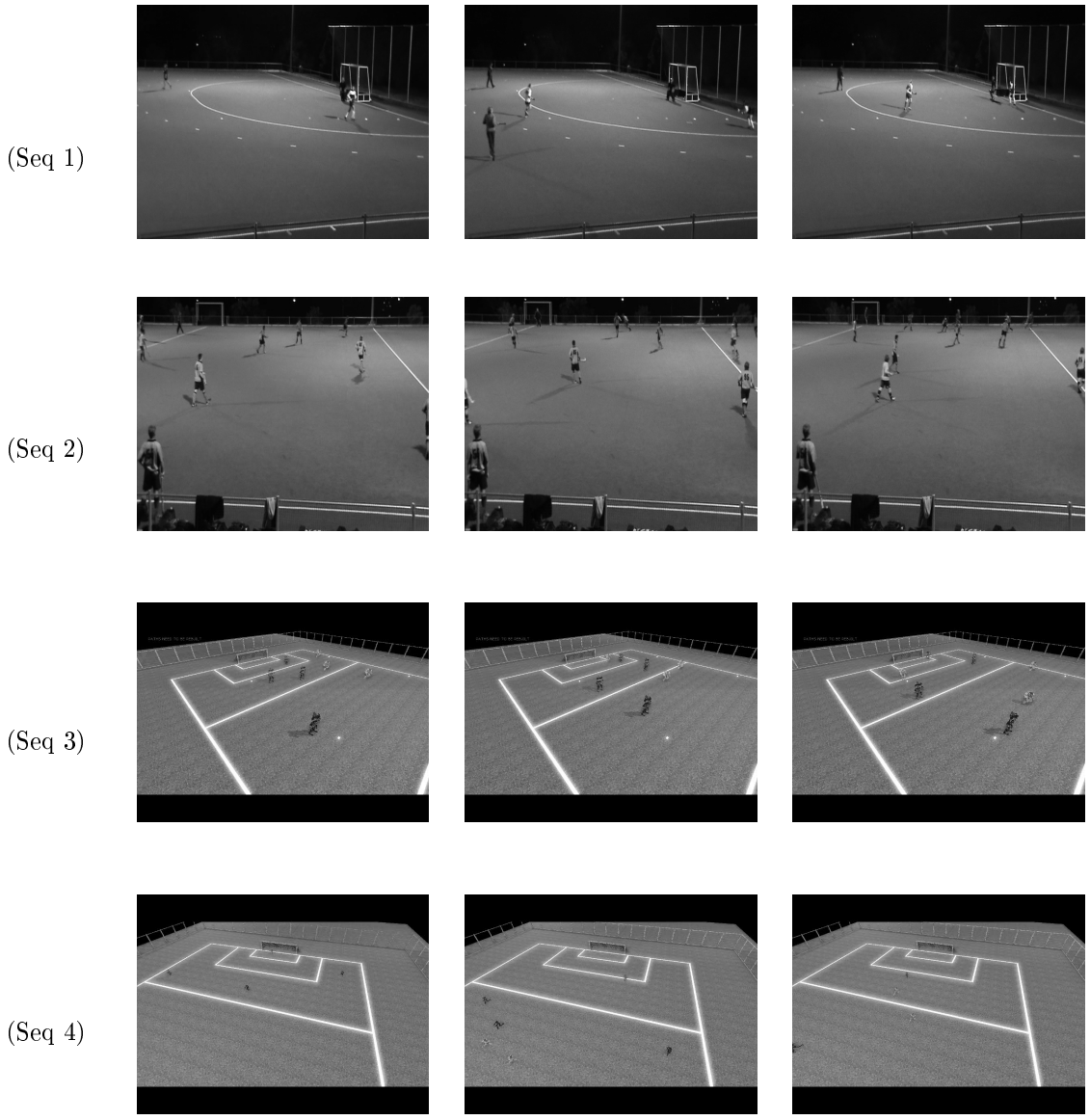


Figure 7.1: Sample frames from motion detection test sequences.

correctly. In the second case it may happen that neither of the players is detected correctly because they appear as one large blob. Figure 7.2 gives an example of each of these two problems. Missed detections can also result from players standing still or moving very slowly for a long period of time.

Figure 7.2: Typical examples of situations causing faulty detections: one player occluding another (left) and two players in close proximity to each other (right).

7.2.2 Passing Detection Results to the Tracker

The experimentally obtained precision and recall of the motion detector, as a stand-alone system, are not bad but not exceptionally good either. However, as mentioned before, it is not that important that a player be detected at the very first possible instance.

Far more crucial is the motion detector's success rate at passing players to the tracker. Recall from section 6.2.1.3 that an object needs to be detected for n frames before eventually being passed

Sequence 1					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.60	0.85	0.94	0.97	0.99
Recall	0.77	0.87	0.84	0.52	0.33

Sequence 2					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.78	0.91	0.94	0.95	0.98
Recall	0.76	0.85	0.77	0.54	0.15

Sequence 3					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.85	0.90	0.91	0.97	0.96
Recall	0.77	0.82	0.71	0.31	0.04

Sequence 4					
	$t = 5$	$t = 8$	$t = 10$	$t = 15$	$t = 20$
Precision	0.94	0.97	0.98	0.98	0.97
Recall	0.92	0.94	0.95	0.97	0.57

Table 7.1: Precision and recall of the motion detection on the video sequences. The threshold t specifies the amount of motion necessary in a region for it to be classified as a moving object.

on. For these experiments we used $n = 5$.

In order to investigate the success of the motion detection stage in the larger tracking system, we measure the following four quantities:

(i) average number of frames taken to pass a new object to the tracker;

(ii) number of true objects missed entirely and never passed to the tracker;

(iii) number of incorrect detections (non-players) that are passed to the tracker;

(iv) number of true objects correctly passed to the tracker.

These measurements are presented in Table 7.2 for the four test video sequences. Table 7.3 lists for each of the players visible in the four sequences the number of frames that she is in view before being passed on to the tracker by the motion detector.

In the first sequence there are no missed or incorrect handovers. The average time taken by the motion detector to hand over players is rather slow but, as is apparent from Table 7.3, this is mainly due to two players. The first of these (a_4) entered the scene in close proximity to another player and only once she moved away from the other player was she picked up by the motion detector as a separate object. The second player with a long detection time (a_5) was the goalkeeper, who stood still for a long time, and was only detected after significant movement.

The missed detection in the second sequence resulted from a player that entered the scene in close proximity to another player, towards the end of the video, and never moved far enough away from the other player for her to be detected as a new player. The incorrect detection in the second sequence came as a result of slight camera movement so that the background model was, for a short

	Sequence 1	Sequence 2	Sequence 3	Sequence 4
(i) avr time per handover	27.3	14.8	5.7	5.7
(ii) missed players	0	1	0	0
(iii) incorrect handovers	0	1	0	0
(iv) correct handovers	6	11	7	7

Table 7.2: Measurements indicating the success of the motion detection system at its primary role of handing over correct objects to the tracker.

Sequence 1						
player	a_1	a_2	a_3	a_4	a_5	a_6
detection time	5	8	19	35	88	9

Sequence 2											
player	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}
detection time	5	5	5	5	5	5	5	5	8	15	100

Sequence 3							
player	c_1	c_2	c_3	c_4	c_5	c_6	c_7
detection time	5	5	5	5	5	6	9

Sequence 4							
player	d_1	d_2	d_3	d_4	d_5	d_6	d_7
detection time	5	5	5	5	5	5	10

Table 7.3: Detection times for all the players in the video sequences. Each is given as the number of frames that the player is actually in view before being passed to the tracker by the motion detector.

period of time, inaccurate. Player b_{11} also entered the scene in close proximity to another player, hence the longer detection time.

Sequences three and four have no missed players, no incorrect handovers and low average detection times. These two sequences illustrate the effectiveness of the detection stage, providing exceptional results under near ideal conditions.

7.2.3 Tracking

Before quantitative results for the tracking stage are presented, a note must be made on the use of UDK videos as opposed to real-life videos. Whilst the UDK videos are useful as the ground truth for them are known, they may not provide full insight into the reliability of the system in real-life situations. For the quantitative results UDK videos are used for their ground truth properties. To illustrate that the tracking system does work for real-life videos, figure 7.3 shows some results for tracking from a video recorded with a hand-held camera. Each instance is a set of 100 frames.

To test the accuracy of the 2D player tracking a single humanoid was tracked through four sequences of 286 frames. For each frame the center of the humanoid was manually annotated. The distance between the center of the tracking block (a blocksize of 80×40 was used) and the annotated

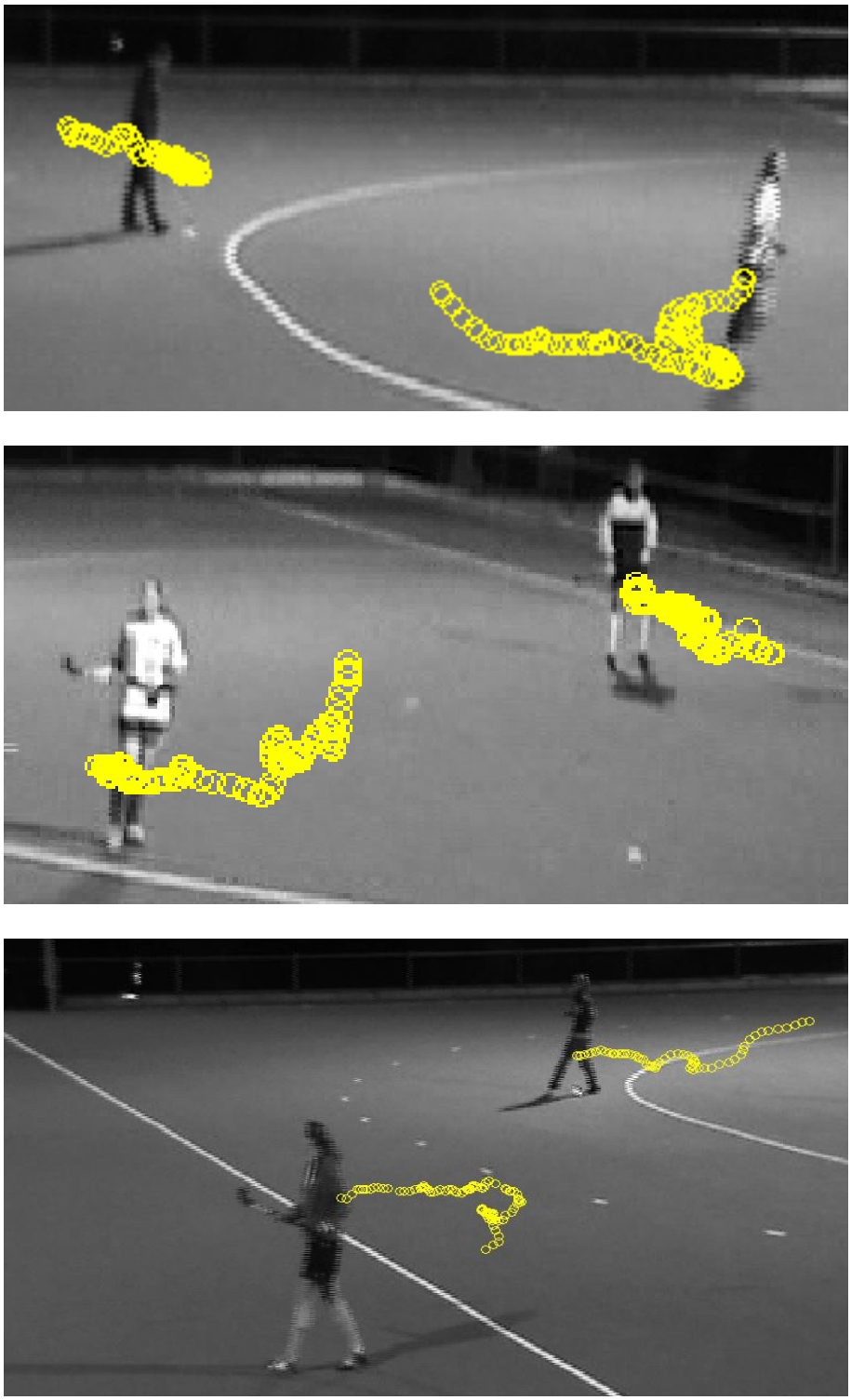


Figure 7.3: Tracking players through a video sequence as recorded by a hand-held video camera.

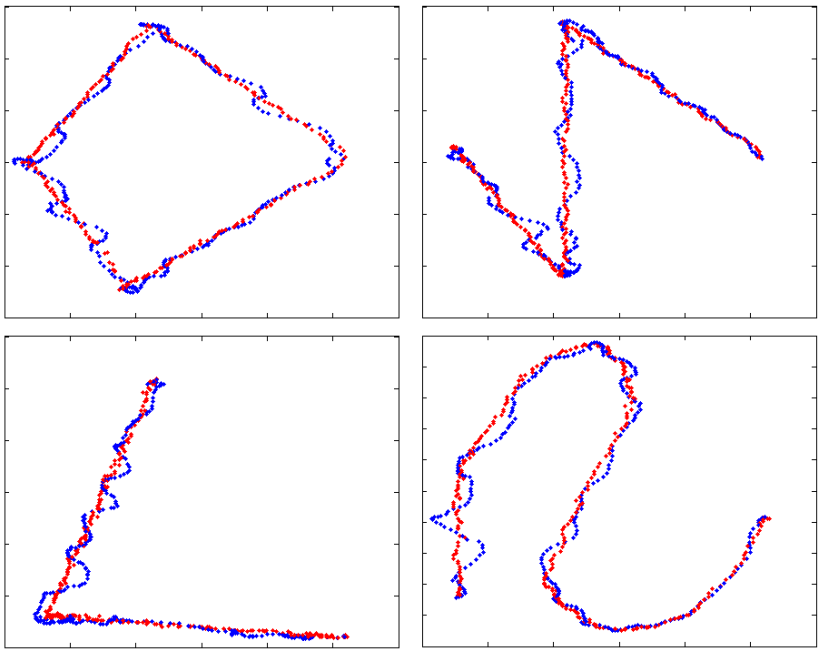


Figure 7.4: Results of tracking a humanoid in 2D through a video sequence. The red points are the manually annotated points and the blue points are the tracking results.

Sequence	Maximum deviation	Average deviation
1	19.5	8.08
2	18.1	6.29
3	25.7	7.67
4	21.4	7.01

Table 7.4: Maximum and average deviation of tracked players measured against manually annotated ground truth positions (measured in pixels).

point was calculated for each frame. Figure 7.4 illustrates the results of the four sequences. The red points are the annotated points and the blue ones the tracking results.

Table 7.4 shows the maximum and average deviation between the tracked and annotated points. The average deviation between the tracked and annotated points ranges between 6.29 and 8.08 pixels in the four sequences. This average deviation result indicates that the tracking system remains fairly close to the actual path for the duration of the sequence. The maximum deviation results may be a cause for some concern. If the deviation exceeds 30 pixels in a purely horizontal direction (or 60 pixels in the vertical) there is a risk of the tracker losing the player. Fortunately the highest maximum deviation found was 25 pixels and this was not in a horizontal direction.

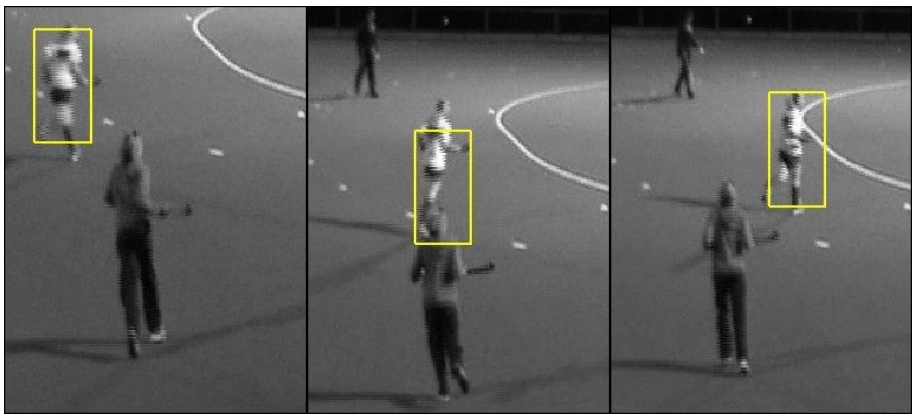


Figure 7.5: Tracking a player through a partial occlusion.

Figure 7.6: Tracking a player through a full occlusion.

Finally the ability of the tracking system to handle occlusions must be discussed. A partial occlusion (from the camera's point of view) occurs when one player covers part of another for some period of time. If the area of occlusion is small so that most of the partially occluded player remains visible, the tracker is able to follow her correctly. Figure 7.5 shows an example before, during and after such a partial occlusion.

In a full occlusion one players obscures another more-or-less completely. Figure 7.6 shows an example. In this case the tracker loses the occluded player and may, after the occlusion when the two players separate, erroneously follow the occluding player. This may happen because (in our current system) template models are updated rather quickly so that, in the event of a full occlusion, both trackers may lock onto the front-most player.

In the following section the tracking system is combined with triangulation to calculate the accuracy of 3D position estimation.

7.2.4 Triangulation

The two triangulation methods, forward and back projection error minimization, are analyzed and compared in two experiments. In the first experiment random points were created and projected onto the image planes of several cameras, with noise added. The points were then triangulated using the noisy projections and the distance between the actual point and the triangulated point were calculated. In the second experiment several synthetic sequences of a humanoid running on a preplanned path were created. The humanoid was then tracked and triangulated from multiple views using the discussed method. Sequences were created using the UDK.

The first experiment will provide insight into the accuracy of each of the projection methods in a controlled artificial environment. The second experiment is done to measure the accuracy of the triangulation methods on typical data that would be encountered when the system is applied to a real match.

For the first experiment a random point within the combined field of view of all the cameras is created. This point is projected onto each camera's image plane and then some noise is added to the projected point. The original point is recalculated using the two triangulation methods. Figure 7.7 shows the results of this experiment. The average distance between the original point and the triangulated point (over $10,000$ iterations) is plotted against the amount of noise added to the projected point. From this figure it can be seen that for low noise situations the triangulation accuracy is similar for the two methods. At higher noise levels the forward projection method performs better than the back projection method, with triangulation results 4% better at projection noise of 50 pixels.

For the second test synthetic sequences were created using UDK. By artificially creating a sequence in UDK it is possible to record a humanoid running on a pre-planned path for which the ground truth is known. This allows for more accurate analysis of the triangulation technique than using real-world data for which ground truth is not known. Figure 7.8 shows the results for four such sequences. The solid blue line in each figure is the ground truth path that the humanoid ran over, as viewed from above. The blue points are the back projection results and the green points are the forward projection results.

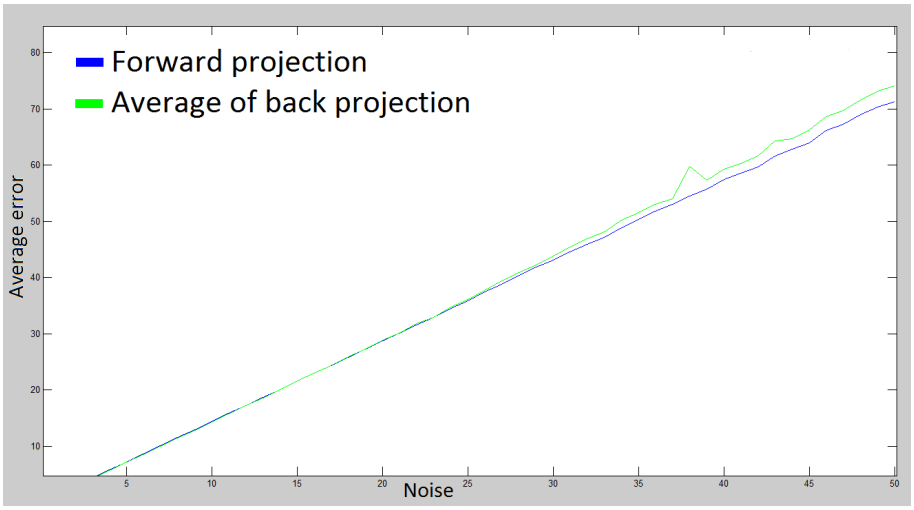


Figure 7.7: Performance comparison of forward and back projection triangulation techniques.

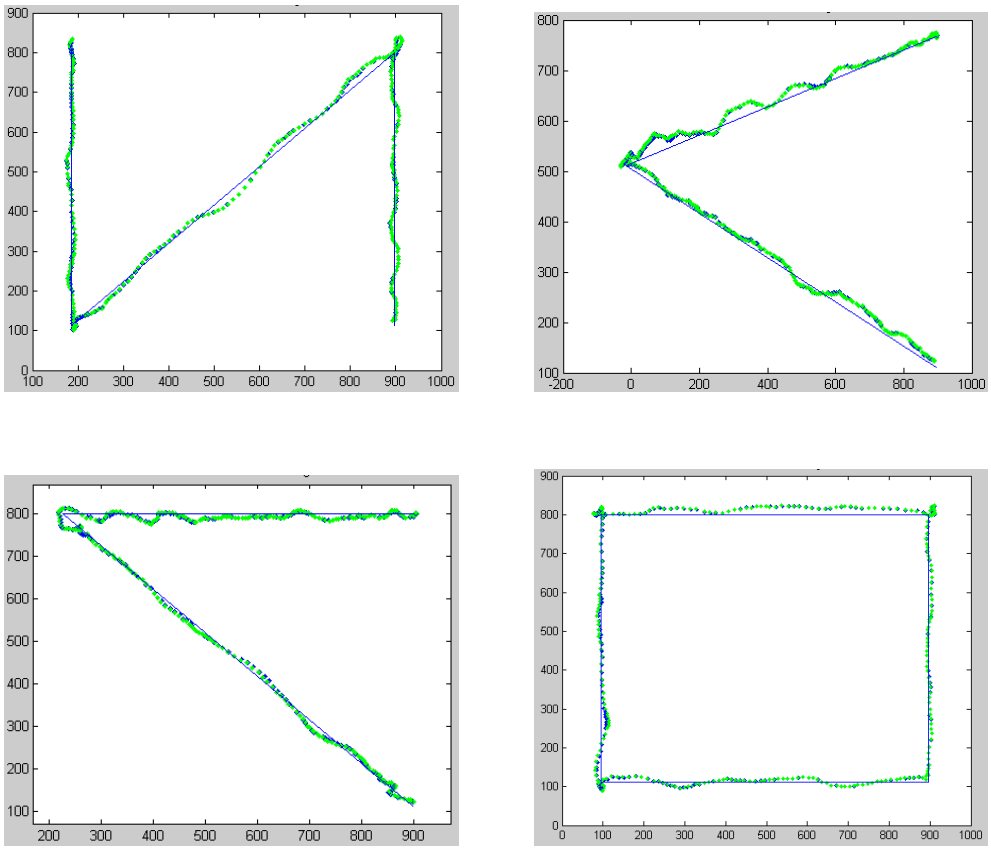


Figure 7.8: Triangulation of two image sequences using forward (green) and backprojection (blue) methods. The solid blue line indicates the ground truth.

This test again indicates that triangulation results for the two methods are similar. This test also indicates that the proposed tracking and triangulation method succeeds at locating a player on the field of play. In the given figures one unit of measurement corresponds to 2 cm on a real-life field. The maximum deviation from the ground truth between the four sequences is 20 units (40 cm) while the average deviation is about 5 units (10 cm).

7.3 System Results

In figure 7.9 the full system results can be seen for tracking 4 players as seen in 4 views over 286 frames (10 seconds). The coloured dots indicate the triangulated position for each player through the sequence, as viewed from directly above the field. The approximate position of the cameras are shown by the small camera drawings. As can be seen from this figure the system as a whole functions as desired: detecting, tracking and triangulating each of the players. This is, however, an ideal case and further testing of some possible problem scenarios needs to be done.

During full system testing two cases of interest were identified. The first case is when a player leaves or enters a camera field of view and the second is when a 2D tracker loses its player due to occlusion.

In figure 7.10 the solid lines indicates field of view boundaries of the different cameras, and the blue dots indicates the path followed by the player. At point **(a)** the player moves out of the view of the camera indicated by the green lines. At this point the corresponding 2D tracker is stopped and the player is triangulated with the remaining views. At point **(b)** the player moves back into the field of view of the camera and is tracked in that view again. Single view position estimation showed that this view corresponded to the player already being tracked and the player is then triangulated using the data from that view as well.

The second case is illustrated in figures 7.11 and 7.12, where in one view the two players move in such a way that the one player occludes the other whilst in the other two views they move apart from each other. In frame (1) the players are some distance away from each other. By frame (10) they have started to occlude each other and at frame (32) they are heavily occluded. As can be seen in frame (48) the tracker indicated by the blue square has begun to track the incorrect player. At

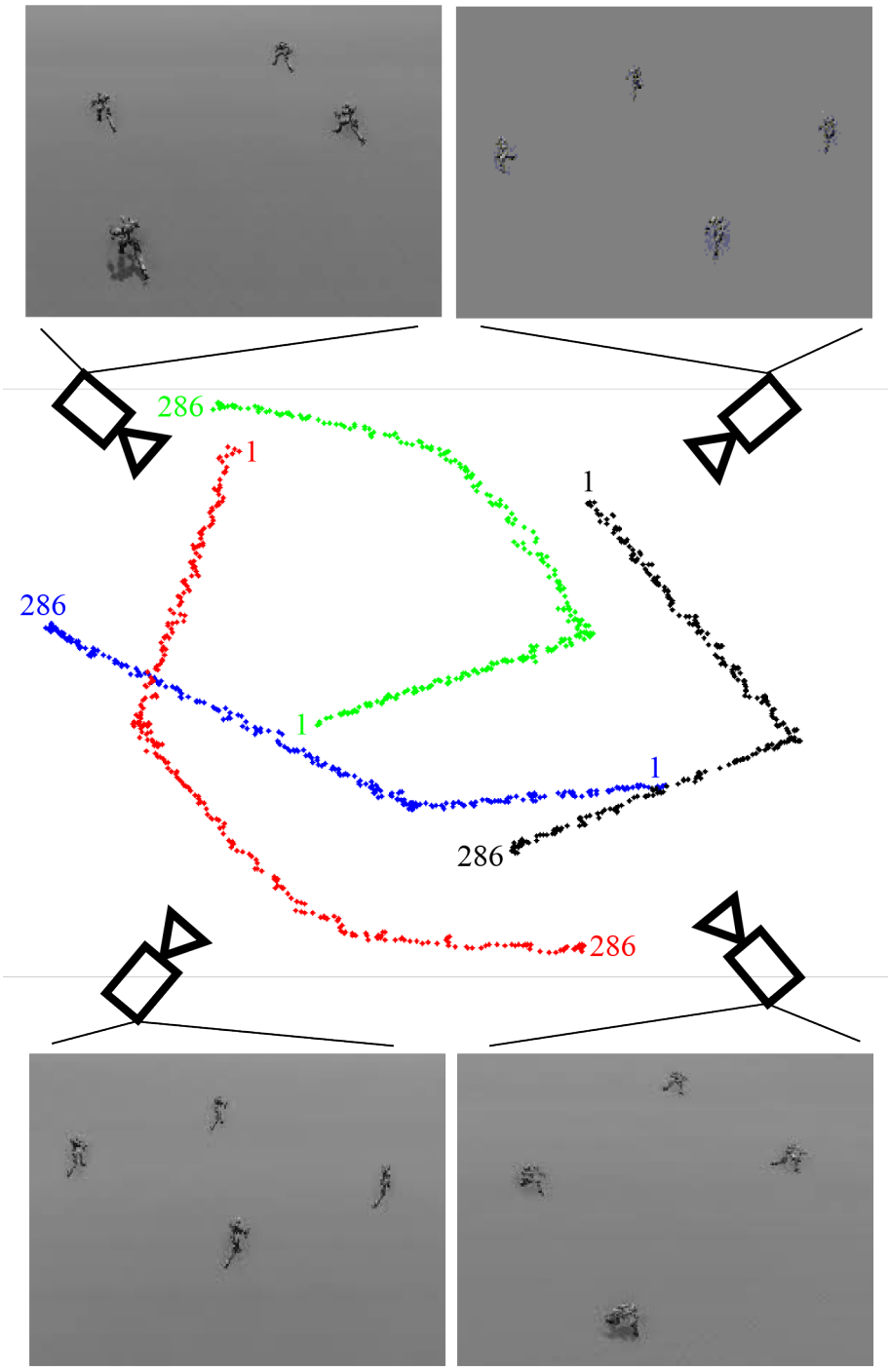


Figure 7.9: Full system results for tracking four players though 286 frames. Positions at first and last frames are noted. Frame 1 is also shown for each camera.

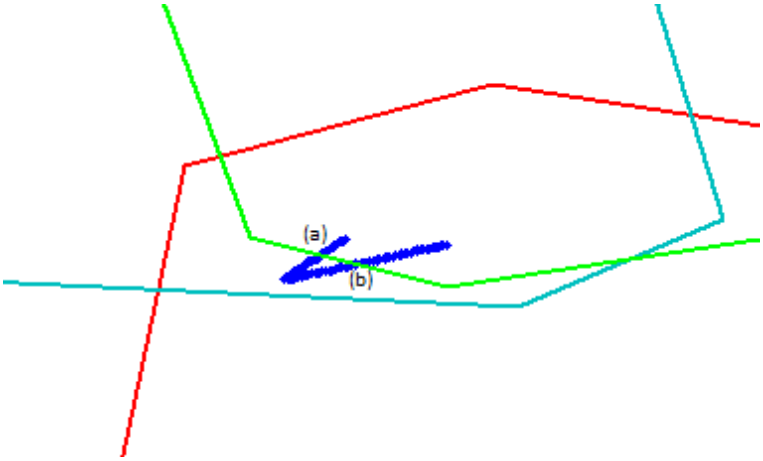


Figure 7.10: Tracking a player crossing the field-of-view line of a camera, shown here from a top down view.

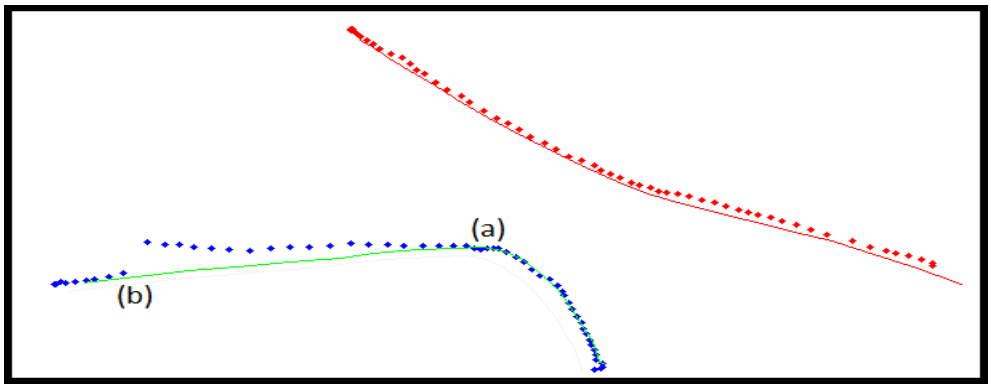


Figure 7.11: Triangulation results of the multi-view tracking in figure 7.12, as viewed from above.

this point the system detected that the 2D tracker has lost the player and moved from the correct path and attempts to correct the mistake. In frame (49) the 2D tracker in the first view has been corrected by back projection after triangulating the player using the rest of the views. By frame (57) the 2D tracker has corrected itself and is tracking the player correctly again.

The plot of the players' positions in figure 7.11 illustrates the effect of this occlusion. At point **(a)** the triangulation result begins to drift from the ground truth line. At point **(b)** the 2D tracker is corrected and the triangulation results snap back to the correct ground truth line.

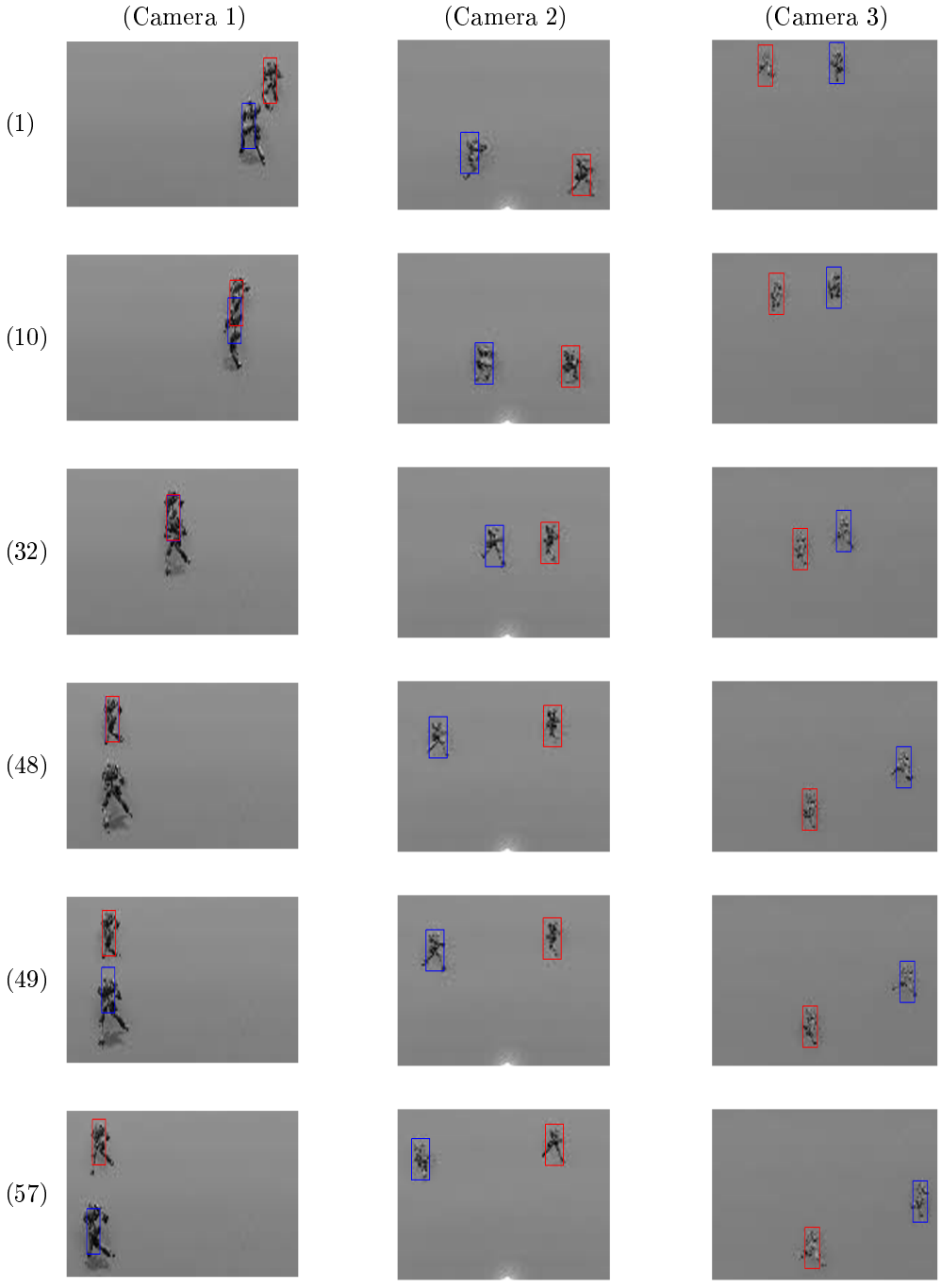


Figure 7.12: Automatic correction of tracking occlusion. Frame numbers are listed on the left of the images.

Chapter 8

Conclusions and Future Work

In this thesis the problem of estimating the positions of players on a sports field was discussed. Past techniques, required components and overall system design were all covered in some detail.

8.1 Conclusions

At the start of the thesis several existing approaches to position estimation of people in a known environment were discussed in methodology and with regards to their strengths and weaknesses. This allowed us to identify possible routes to follow and areas to focus on.

After having discussed previous attempts at solving the problem of position estimation the various components required to build a complete system were discussed. By comparing the different components it was possible to decide on a subset of them that would provide an effective solution to the stated problem. Motion detection proved to be a simple, effective player detection solution and the particle filter was found to be a useful tool for 2D tracking. Accurate multi-view triangulation that shares computation with previous steps was chosen for position estimation.

In chapter 6 the various components were combined to form a complete system that is able to detect and provide position estimates for players on a sports field. Some algorithms discussed earlier were expanded upon and feedback between different stages of the systems was introduced to increase

accuracy. This system was analyzed in chapter 7 to determine if it is capable of solving the given problem.

The results obtained for the system were very promising overall. While some improvements can be made the system is able to solve the initial problem to a satisfactory extent. The lesser goal of real-time processing was not achieved during implementation, but the use of a graphical processing unit (GPU) along with some algorithm optimization should achieve this goal with relative ease.

8.2 Future Work

As stated in the previous section, there is some room to improve on the system to increase the accuracy of the results. Two of these areas would be to remove the fixed block size during tracking and improve 2D occlusion handling.

The fixed block size used during tracking provides a minor problem during tracking as the player size on the image may vary depending on how far away they are from the camera. This causes the tracker to lose accuracy in the 2D stage, which propagates to less accurate triangulation results. Allowing the block size to vary would go some distance to improve this problem. One method would be to vary size of the block as a function of the position of the player in the image, since each camera is calibrated to the playing field.

The introduction of multiple cameras gives the system some robustness against incorrect 2D tracking during occlusions. However solving the problem in the 2D stage will provide better results overall. Some work on this stage to allow 2D trackers to correctly track through occlusions without a feedback loop from the triangulation step would be of great aid. This is a difficult problem, however, which would require a more accurate motion model of human movement in the particle filter. Since people, especially sports players, can move erratically and unpredictably such a model can be hard to specify.

Adding a player recognition component to the system may increase the accuracy, as well as the useability of the system. Some feature recognition algorithm can be trained to recognise the players by using footage of previous games. If this is done, 2D trackers can be grouped based on whom they are tracking without the need for a ground plane matching stage. Such a system may also allow the

system to be implemented for contact sports such as rugby. In the current system, an event such as a ruck or a maul would likely cause most of the 2D trackers to lose their players. This in turn will cause the 3D tracking to fail, and the 2D trackers will be stopped. As the players move apart they will be found as new players by the motion detector and be tracked again. If a recognition stage is implemented the system will be aware that they are the same players as before, allowing the system to continue to track them.

Using the data provided by this system to perform sports analysis could also provide some areas for future work. Post processing or filtering of the output data might be required to provide a more accurate estimate of how far a player travelled during the game as the output data may give a much longer distance than the ground truth, as the estimated path typically forms a wavy line around the ground truth path.

The work discussed in this thesis can be applied to many real-life situations. The main focus was on sports applications, but with some modification the system may be used for security or logistic purposes as well. With cameras and computers becoming more and more accessible and taking an ever increasing role in our lives, such research into how they can be used to improve or aid real-life situations is becoming all the more important.

Appendix A

QR Factorization

QR factorization is an important tool used in linear algebra. It is especially useful for solving linear least squares problems. The factorization decomposes a matrix $\mathbf{A}$ into two matrices $\mathbf{Q}$ and $\mathbf{R}$, where $\mathbf{Q}$ is orthogonal and $\mathbf{R}$ is an upper triangular matrix. To find the QR decomposition we use the three Givens rotations:

$$\mathbf{G}_1 = \begin{bmatrix} c & 0 & s \\ 0 & 1 & 0 \\ -s & 0 & c \end{bmatrix},$$

$$\mathbf{G}_2 = \begin{bmatrix} c & s & 0 \\ -s & c & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{G}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & s \\ 0 & -s & c \end{bmatrix},$$

where $c = \cos(\theta)$ and $s = \sin(\theta)$. First we left multiply $\mathbf{A}$ by $\mathbf{G}_1$, with $\theta = \arctan(\frac{A_{3,1}}{A_{1,1}})$, to set $\mathbf{A}_{3,1}$ to zero. Next we use $\mathbf{G}_2$ and $\mathbf{G}_3$ respectively to set $\mathbf{A}_{2,1}$ and $\mathbf{A}_{3,2}$ to zero. We are now left

with an upper triangular matrix $\mathbf{R}$ so that

$$\mathbf{G}_3\mathbf{G}_2\mathbf{G}_1\mathbf{A} = \mathbf{R}. \tag{A.1}$$

The $\mathbf{Q}$ matrix is obtained by combining the three Givens rotations with $\mathbf{Q}^T = \mathbf{G}_3\mathbf{G}_2\mathbf{G}_1$. This gives us:

$$\mathbf{Q}^T\mathbf{A} = \mathbf{R}, \tag{A.2}$$

so that

$$\mathbf{A} = \mathbf{QR}. \tag{A.3}$$

Appendix B

Singular Value Decomposition

Singular value decomposition (SVD) takes a rectangular $n \times p$ matrix, $\mathbf{A}$, and decomposes it into three matrices:

$$\mathbf{A} = \mathbf{USV}^T, \tag{B.1}$$

where $\mathbf{U}$ is $n \times n$, $\mathbf{S}$ is $n \times p$ and $\mathbf{V}$ is $p \times p$. Also

$$\mathbf{U}^T\mathbf{U} = \mathbf{I} \tag{B.2}$$

$$\mathbf{V}^T\mathbf{V} = \mathbf{I} \tag{B.3}$$

i.e. $\mathbf{U}$ and $\mathbf{V}$ are orthogonal, and $\mathbf{S}$ contains the singular values of $\mathbf{A}$ along its main diagonal.

To calculate the SVD we multiply (B.1) from the left and the right by $\mathbf{A}^T$ respectively. This gives us:

$$\begin{aligned}
\mathbf{AA}^T &= \mathbf{U}S\mathbf{V}^T\mathbf{A}^T \\
&= \mathbf{USV}^T[\mathbf{U}S\mathbf{V}^T]^T \\
&= \mathbf{USV}^T\mathbf{VS}^T\mathbf{U}^T \\
&= \mathbf{USS}^T\mathbf{U}^T \\
\mathbf{AA}^T\mathbf{U} &= \mathbf{USS}^T\mathbf{U}^T\mathbf{U} \\
\mathbf{AA}^T\mathbf{U} &= \mathbf{USS}^T
\end{aligned} \tag{B.4}$$

and following a similar argument

$$\mathbf{A}^T\mathbf{AV} = \mathbf{VS}^T\mathbf{S}. \tag{B.5}$$

From this one can see that the columns of $\mathbf{U}$ correspond to the eigenvectors of the symmetric matrix $\mathbf{AA}^T$ and the columns of $\mathbf{V}$ to the eigenvectors of $\mathbf{A}^T\mathbf{A}$. The singular values contained in $\mathbf{S}$ are the positive square roots of the eigenvalues of either $\mathbf{AA}^T$ or $\mathbf{A}^T\mathbf{A}$. As both $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$ are symmetric matrices we know that their eigenvalues are always real.

Bibliography

[1] W. Adams, P. Loustaunau, "An Introduction to Gröbner Bases", American Mathematical Society, Graduate Studies in Mathematics, vol. 3, 1994.

[2] A. Alahi, Y. Boursier, L. Jacques, P. Vandergheynst, "Sport players detection and tracking with a mixed network of planar and omnidirectional cameras", IEEE International Conference on Distributed Smart Cameras, pp. 1–8, 2009.

[3] G. Antonini, M. Bierlaire, M. Weber, "Discrete choice models of pedestrian walking behavior", Transportation Research part B, vol. 40, pp. 667–687, 2006.

[4] M. Arulampalam, N. Gordon, B. Ristic, "Beyond the Kalman Filter: Particle Filters for Tracking Applications", Artech House, 2004.

[5] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions", Bulletin of the Calcutta Mathematical Society 35, pp. 99–109, 1943.

[6] G. Bradski, A. Kaehler, "Learning OpenCV: Computer Vision with the OpenCV Library", O'Reilly Media, 2008.

[7] M. Byrod, K. Josephson, K. Astrom, "Fast optimal three view triangulation", Asian Conference on Computer Vision, pp. 549–559, 2007.

[8] Q. Cai, J. Aggarwal, "Tracking human motion using multiple cameras", International Conference on Pattern Recognition, vol. 3, pp. 68–72, 1996.

[9] T. Camus, "Real-time quantized optical flow", Journal of Real-time Imaging, vol. 3, pp. 71–86, 1997.

[10] J. Canny, "A computational approach to edge detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, pp. 679–714, 1986.

[11] S. Cha, S. Srihari, "On measuring the distance between histograms", Pattern Recognition, vol. 35, pp. 1355–1370, 2002.

[12] S. Choi, Y. Seo, H. Kim, K. Hong, "Where are the ball and players: soccer game analysis with color-based tracking and image mosaick", Image Analysis and Processing, pp. 196–203, 1997.

[13] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection," IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886–893, 2005.

[14] E. Dougherty, J. Astola, "Nonlinear Filters for Image Processing", IEEE Computer Society Press, 1999.

[15] R. Duda, P. Hart, "Use of the Hough transformation to detect lines and curves in pictures", Communications of the ACM, vol. 15, pp. 11–15, 1972.

[16] A. Elgammal, D. Harwood, L. Davis, "Non-parametric model for background subtraction", European Conference on Computer Vision, vol. 2, pp. 751–767, 2000.

[17] K. Engel, "Real-time Volume Graphics", A. K. Peters Ltd, 2006.

[18] W. Freeman, M. Roth, "Orientation histograms for hand gesture recognition", International Workshop on Automatic Face and Gesture Recognition, pp. 296–301, 1994.

[19] Y. Freund, R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting", European Conference on Computational Learning Theory, pp. 23–37, 1995.

[20] P. Gabriel, J. Verly, J. Piater, A. Genon, "The state of the art in multiple object tracking under occlusion in video sequences", Advanced Concepts for Intelligent Vision Systems, pp. 166–173, 2003.

[21] S. Ganapathy, "Decomposition of transformation matrices for robot vision", IEEE International Conference on Robotics and Automation, pp. 130–139, 1984.

[22] E. Grafarend, P. Lohse, B. Schaffrin, "Dreidimensionaler rückwärtsschnitt teil I: die projektiven gleichungen", Zeitschrift fuer Vermessungswesen, pp. 1–37, 1989.

[23] J. Grunert, "Das pothenotische problem in erweiterter gestalt nebst uber seine anwendungen in der geodasie", Grunert's Archiv fuer Mathematik und Physik, pp. 238–248, 1841.

[24] A. Haar, "Zur theorie der orhofonalen funktionensysteme", Mathematische Annalen 69, vol. 3, pp. 331–371, 1910.

[25] M. Haralick, C. Lee, K. Ottenberg, M. Nolle, "Review and analysis of solutions of the three point perspective pose estimation problem", International Journal of Computer Vision, vol. 13, pp. 331–356, 1994.

[26] R. Hartley, A. Zisserman, "Multiple View Geometry in Computer Vision", 2nd edition, Cambridge University Press, 2003.

[27] M. Hoffmann, K. Hunter, B. Herbst, "The hitchhiker's guide to the particle filter", Proceedings of the 19th Symposium of the Pattern Recognition Association of South Africa, pp. 33–38, 2008.

[28] B. Horn, B. Schunck, "Determining optical flow", AI Memo 572, Massachusetts Institute of Technology, 1980.

[29] T. Horprasert, D. Harwood, L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection", ICCV Frame Rate Workshop, pp. 1–19, 1999.

[30] M. Isard, A. Blake, "Condensation - conditional density propagation for visual tracking", International Journal of Computer Vision, vol. 1, pp. 5–28, 1998.

[31] A. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall International, 1989.

[32] S. Khan, O. Javed, Z. Rasheed, M. Shah, "Human tracking in multiple cameras", IEEE International Conference on Computer Vision, vol. 1, pp. 331–336, 2001.

[33] S. Kullback, R. Leibler, "On information and sufficiency", Annals of Mathematical Statistics 22, pp. 79–86, 1951.

[34] S. Lee, J. Choi, "Correction of radial distortion using a planar checkerboard pattern and its image", IEEE Transactions on Consumer Electronics, pp. 27–33, 2009.

[35] D. Lowe, "Object recognition from local scale-invariant features," International Conference on Computer Vision, pp. 1150–1157, 1999.

[36] D. Lucas, T. Kanade, "An iterative image registration technique with an application to stereo vision", Imaging Understanding Workshop, pp. 121–130, 1981.

[37] K. Mikolajczyk, C. Schmid, A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors", European Conference on Computer Vision, vol. 1, pp. 69–81, 2004.

[38] M. Mozerov, A . Amato, M. Al Haj, J. Gonzalez, "A simple method of multiple camera calibration for the joint top view projection", Computer Recognition Systems 2, vol. 45, pp. 164–170, 2007.

[39] C. Needham, R. Boyle, "Tracking multiple sports players through occlusion, congestion and scale", British Machine Vision Conference, pp. 93–102, 2001.

[40] Y. Ohno, J. Miura, Y. Shirai, "Tracking players and estimation of the 3D position of a ball in soccer games", International Conference on Pattern Recognition, vol. 1, pp. 145–148, 2000.

[41] C. Papageorgiou, T. Poggio, "A trainable system for object detection", International Journal of Computer Vision, vol. 1, pp. 15–33, 2000.

[42] B. Profitt, "Background subtraction algorithms for a video based system", MSc Thesis, Stellenbosch University, 2009.

[43] Y. Rubner, C. Tomasi, L. Guibas, "A metric for distributions with applications to image databases", International Conference on Computer Vision, pp. 59–66, 1998.

[44] H. Schneiderman, T. Kanade, "A statistical method for 3D object detection applied to faces and cars," IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 1746–1759, 2000.

[45] D. Simon, "Kalman filtering", Embedded Systems Programming, vol. 14, pp. 38–53, 2006.

[46] X. Song, J. Cui, H. Zha, H. Zhao, "Probabilistic detection-based particle filter for multi-target tracking", British Machine Vision Conference, pp. 223–232, 2008.

[47] C. Stauffer, W. Grimson, "Adaptive background mixture models for real-time tracking." IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 246–252, 1999.

[48] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses", IEEE Journal of Robotics and Automation, vol. 4, pp. 323–344, 1987.

[49] V. Vapnik, "The Nature of Statistical Learning Theory", Springer-Verlag New York, 1995.

[50] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 511–518, 2001.

[51] R. Vos, W. Brink, "Combining motion detection and hierarchical particle filter tracking in a multi-player sports environment", Proceedings of the 20th Symposium of the Pattern Recognition Association of South Africa, pp. 65–70, 2009.

[52] H. Wang, D. Suter, "A re-evaluation of mixture-of-Gaussian background modeling", IEEE International Conference on Accoustics, Speech and Signal Processing, vol. 2, pp. 1017–1020, 2005.

[53] G. Welch, G, Bishop, "An introduction to the Kalman filter", Technical Report TR95-041, University of North Carolina, 1995.

[54] M. Xu, J. Orwell, L. Lowey, D. Thirde, "Architecture and algorithms for tracking football players with multiple cameras", Intelligent Distributed Surveilliance Systems, pp. 51–55, 2004.

[55] A. Yamada, Y. Shirai, J. Miura, "Tracking players and a ball in video image sequence and estimating camera parameters for 3D interpretation of soccer games", International Conference on Pattern Recognition, vol. 1, pp. 303–306, 2002.

[56] C. Yang, R. Duraiswami, L. Davis, "Fast multiple object tracking via a hierarchical particle filter", International Conference on Computer Vision, vol. 1, pp. 212–219, 2005.

[57] C. Yu, Q. Peng, "Robust recognition of checkerboard pattern for camera calibration", Optical Engineering, Vol. 45, 2006.

[58] Z. Zivkovis, "Improved adaptive Gaussian mixture model for background subtraction", International Conference on Pattern Recognition, vol. 2, pp. 28–31, 2004.

[59] http://www.doh.gov.za/facts/1998/sadhs98/chapter13.pdf

[60] http://www.UDK.com/

[61] http://www.wegame.com/