Universiteit Utrecht

Master of Science Thesis

Extracting ground plane location data of rugby 7s players, from a zooming and rotating camera

J. van Dis B.Sc.

February 20, 2016

In collaboration with

Faculty of Science · Universiteit Utrecht

Extracting ground plane location data of rugby 7s players, from a zooming and rotating camera

Master of Science Thesis

For obtaining the degree of Master of Science in Technical Artificial Intelligence at Utrecht University

J. van Dis B.Sc.

February 20, 2016

Supervisors:

dr. ir. R.W. Poppe	Universiteit Utrecht
dr. F.P.M. Dignum	Universiteit Utrecht
J. Heizenberg	Capgemini B.V.
F. Burgers	Capgemini B.V.

Faculty of Science · Universiteit Utrecht

Universiteit Utrecht

Abstract

Finding the trajectory of sports persons in videos can be done in various ways. We address this problem for rugby 7s under difficult conditions, such as occlusion, a small camera angle with respect to the horizon, zooming, and supporters standing nearly in the field. A methodology is presented that processes videos of rugby 7s games during which simultaneously the camera parameters are derived - stating its location, orientation and internal settings such as focal length, and the position of the rugby players within the image plane are detected, to tracks their movements with respect to the ground plane. We propose a new method to determine the camera parameters effectively, making use of the recurring nature of the camera state, by only using a limited set of reference frames, called baseframes. Tracking is performed in the ground plane, using a tracking-by-detection approach with a Kalman filter to dynamically model the tracklets. An evaluation of our methodology shows the potential of our methodology. However, it is recommend to improve of the conditions under which tracking is performed.

Acknowledgements

A little over a year ago, as I first stepped into the office of Ronald Poppe, I asked him whether he knew any *nice* projects at companies within the Computer Vision industry that would be fitting for a master thesis. A month later, I came into contact with Capgemini and in February last year, I began working on this project.

Ronald has been giving me tons of feedback, as a supervisor, throughout the project, and helped me out whenever I had questions - even with the obvious ones. The several iterations of me handing in another version and him giving me his critique really allowed me to make the best out of this document.

Kevin Gallagher supported me around the beginning of the project, allowing me some insight in the operations of data analysis within the NRB. He gave me advice on how videos were taken, and what possible approaches for my project would be, for which I'm grateful. At Capgemini, Fedde Burgers and Erik van Berkel helped me during the project, by allowing me to pitch some of my ideas to them. Fedde, as my official supervisor there, was able to share his insights as well as his graduation work on action recognition. Although not completely related to this project, it helped to had someone to talk 'computer vision' with at Capgemini.

During my thesis, I have been thinking much about my role in professional life would be after I graduate, which hasn't always been easy. A big thanks goes to Johan van Houten, for his support during these times. He was able to give me focus and direction towards my goals and aspirations. I also thank the members of my band, Homemade Water; Laurens, Eline, Andrea and Moos. I'd like to think that the music we made last year, boosted me up and motivated me to keep working on this project.

Lastly, I would like to thank Elisabeth, for her endless support throughout this enduring project. Her sometimes blunt statements about me being too detailed, or focussing on the wrong elements made me laugh and helped me keeping on track with the things I needed to do. She put up with me this last year, and she strongly encouraged me to really finish my thesis.

Utrecht, The Netherlands
February 20, 2016

J. van Dis B.Sc.

Contents

Chapter 1

Introduction

Video tracking is defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene [36]. Ground plane tracking is estimating the trajectory of an object relative to a self defined ground plane, such as the physical ground plane. Numerous approaches and techniques have been proposed in the past, such as particle filters, mean shift trackers, KLT, and body part trackers. The techniques are often used with static or semi-static cameras - cameras which cannot move or or only allowed to rotate. The problem becomes more difficult when a camera is allowed to zoom, and thus changing the relation between a location in the 3D world and the 2D image.

Until recently, the Dutch Rugby Women's Sevens team were qualifying for the Olympic games of 2016 in Rio de Janeiro. The Dutch Rugby Federation (NRB) asked IT consultancy company Capgemini Nederland B.V. to aid them in their search for statistical performance data. In order to gain more insight in the training process, as well as to improve the individual and team performances, feedback is required. It is believed that giving the right feedback and allowing the right insights to the rugby coach and his team, the performance of the rugby women their team will improve.

Initially the analyses were performed on data from heart monitors, which each player got to wear. The analyses tried to track, amongst others, the perceived intensity of trainings and matches. But recently the need has emerged for additional insights. This started off with the introduction of location data from GPS to perform analyses. With the availability of lots of video data, the NRB was open to the idea of deriving statistical information from videos of matches played by their national team. When location data of two competing teams is available, a more detailed analysis can be performed, both on an individual basis as well as on a team basis. For example, feedback on guarding and positional play can be given. Therefore, location data of the players of both teams is desirable. The GPS data,

which is available in small amounts, is used as both an initialization and validation tool. A first objective has been set to derive the location data of all players on the field throughout the match.

The goal of this report is to present a methodology that allows for robust tracking of rugby players with the use of video material. The methodology comes in two distinct parts: (1) the derivation of the camera parameters stating its location, orientation and internal settings such as focal length, and (2) the detection and tracking of rugby player locations from the image plane, translated and projected onto the ground plane.

The remainder of this report is structured as follows. It continues with a detailed description of the aspects of the problem in Chapter 2, where we present our research question and sub questions. The requirements, the assumptions as well as the available data are described and discussed, as well the scope and relevance of our work. A chapter on background techniques and related work is presented in Chapter 3. Chapter 4 contains a detailed description of our methodology; the various design choices are explained and a detailed description of the camera parameter estimation and player tracking are given. In Chapter 5, on experiments and results, an implementation is shown of the proposed method together with a series of experiments. Chapter 6 concludes this report and gives recommendations on future improvements and extensions.

Chapter 2

Project definition

For reviewing purposes, all trainings and matches of the Dutch National Rugby team are recorded with a video camera. With a renewed, extended interest in statistics and insights, it has been requested to perform analysis on the available video data of the matches. It is believed this will give insights in, amongst others, player workload, tackle success rate and scoring strategies. A first objective is set to derive accurate ground plane location data of all the players on the field. This chapters explains the details of the problem at hand, including the available data and the requirements. It also describes the relevance of this project in general, and to science.

2.1 Problem description analysis

There is a need for ground plane location data of rugby players during a match. The location data is requested for all *active* players on the field, from both the Dutch and the opponent team. This will allow for more elaborate and complete analyses in the follow-up study, which is concerned with the analysis of the location data.

The ground plane location data is to be extracted from the videos provided by the Dutch Rugby Association (NRB) during a tournament in Las Vegas, USA. These videos are recorded with a hand-held camera on a tripod, and contain zooming and rotation, but no translation. In addition to the video data, a small amount of GPS data, which was recorded by Johan-sports, is also made available. This gives spatio-temporal data - (GPS) location data over time - of the Dutch players during the matches of the tournament.

Given the videos, there needs to be an estimation of the player locations in the ground plane over time. For this to be possible, there needs to be a tracking algorithm that tracks

the players with respect to a reference frame, and a simultaneous estimation of the camera parameters as well. The detections are mapped to the 3D, onto the ground plane, which serve as input for the tracking part.

In consideration with the data analyst of the rugby association, as well as with Capgemini Nederland B.V., a list of requirements has been built and analysed. These requirements are listed below, and entail both system and data requirements.

REQ.1 For each video, give 2-dimensional ground plane position data of every participating rugby player at least every 200ms. The position needs to be relative the the center of the field.
REQ.2 Deliver a complete methodology and implementation to determine the camera's state (intrinsics, rotation, and location) at every frame using computer vision techniques.
REQ.3 The final model has to give robust results (in contrast to accurate).
REQ.4 The analysis does not have to be performed in real-time.
REQ.5 The target videos should concern a rugby 7s match in which two teams compete, where the teams are distinguishable by colour.
REQ.6 The videos should be filmed with a camera having a single focus point
REQ.7 There is no translation of the camera, only rotation and zooming.
REQ.8 The camera changes are smooth and overlap between subsequent frames should be at least 95%
REQ.9 Calibration videos for the camera are available
REQ.10 There should be GPS data available of at least the first 10 seconds prior to the kick-off.

REQ.1 states that our method needs to give an (X, Y) coordinate with respect to the ground plane, and it is essentially the main goal of our project. It requests a data file for each video, in which for every player a 2D ground plane coordinate should be reported with an interval of at most 200ms. This time requirement has been set in coherence with Johan-sports, who perform data analyses on location data, and report there is no significant increase in data analysis results when the interval drops below 200ms.

The second requirement (REQ.2) goes more into depth about the relation of the camera its view and the world. The field of view of the camera is limited, hence it will occur that some players will not be in the field of view of the camera. As a consequence, these players cannot be actively tracked, and their location data needs to be estimated for the time they are outside the view.

REQ.3 refers to the usability of the resulting data. In order to perform analyses on the location data of players, these location data files should be complete and correct. The last requirement, stating that the analysis does not have to be done in real-time allows us to use computer vision and tracking techniques that deliver results of higher quality, without the need for extreme computing power, as well as backward tracking and post processing. We propose a method that uses forward tracking only, but there is put no constraint on the amount of processing time per frame.

The camera motion should be smooth, is what scope item REQ.8 tells us. Our solution will be based upon a video in which movement follows a natural approach, such that subsequent

frames have a large overlap. Camera calibration videos are essential as to determine the relation between the image coordinates and the camera's coordinate system. It also allows us to derive the relations between the zoom value and the camera parameters. The GPS data is primarily used to determine the dynamics of the players in rugby 7s. Additionally, we use it to initialize it is used to initialise the camera parameters at the start of the video. This is an essential step, as is explained in Section 4.2.4.

The methodology that is created on the basis of the available data. Nevertheless, our implementation and methodology can possibly be used for other sports or similar data as well.In this report, analyses of poses (pose estimation) or action recognition is not included. It is considered outside the scope of this project. The same goes for the analysis of the location data resulting from our implementation.

2.2 Available data

The NRB has a large amount of video data available. Most of the trainings and matches are recorded by a camera for various purposes. Primarily these recordings are used for direct review by the players themselves to identify their movements and analyse their behaviour. The task at hand will serve as a stepping stone to allow this analysis to be done not by hand but automatically. For the project at hand the following data has been made available: video footage of 5 matches, played by the Dutch team during an international tournament in Las Vegas, January 2015. Along with the video data there is GPS data available of the Dutch players, which are provided by Johan-sports. We discuses both types of data subsequently.

2.2.1 Video footage

Each of the videos shows the entire match, and is of approximately 18 to 20 minutes length. The videos have been recorded with a handy cam from the stands, where the camera is placed on a tripod. The camera continuously follows the ball around, and repeatedly zooms in and out to focus on certain actions such as conversions and fouls. Because the video files contain zoom the video has a small number of frames in which the camera is out of focus.

The field itself is hard to see in the videos, mainly because of the field's bad quality. There are large areas on the fields which contain just sand, and the field lines are a dark tint or red. The players however are easily identifiable, especially since the orange shirts of the Dutch form a high contrast with respect to the background. The background is rather cluttered, with a second match being played on the field behind the one of interest. Some example frames of the videos can be seen in Figure 2.1.

The camera used to record the matches has analogue image stabilization, but the video files contain no trace of digital deformation or post-processing. The video frames are of size 1080×720 pixels, and the videos have a frame rate of 25 frames per second.

2.2.2 GPS data of the Dutch team

For each of the available videos, there is a corresponding set of GPS data available of the Dutch players provided by the Dutch company Johan-sports [14]. During the matches the

Figure 2.1: Example frames from the available video data set. It clearly shows the bad state of the field, as well as the high contrast of the players with respect to the background.

players wore GPS trackers at the chest area, from which measurements were performed at a frequency of 5 Hz. This GPS data is sent to a general storage server, and assembled into a single file.

The dataset received from Johan-sports is pre-processed, where the location data has been converted from GPS coordinates to distances with respect to the centre of the field during post-recording. They used the GPS coordinates of the corners of the field to convert it. A small portion of the GPS data is shown in Table 2.1. The two time columns are defined as follows. The first one is the time in seconds, with base January 1st, 1970 (00:00). The second time, in milliseconds, is based on the moment the GPS receiver was turned on.

2.3 Research question

The following research question is formulated:

Given a video of a Rugby 7s match filmed by a camera placed on a tripod along the side line, to what extent is it possible to give 2D ground plane location data through time of all players on the field during that match, throughout an entire match from start to finish, up to at least 5 locations per seconds?

The research question encompasses the requirements, and bounds the objective to the data that has been made available. In order to make this question more tangible, a number of sub questions have been established, that will aid in answering the research question:

1. What are the camera parameters throughout the video? Give orientation, position, and intrinsic camera settings (focal length, principal point, distortion) for every frame.
2. Up to what precision and accuracy can players automatically be detected in video frames using computer vision techniques?

Table 2.1: The two timestamps combined form the actual time. The base of the time in seconds is January 1, 1970. The base of the second column is when the GPS receiver was turned on.

time [s]	time[ms]	id	x-pos[m]	y-pos[m]
1423848231	1774530	1	27.58084626	12.48146965
1423848231	1690961	2	27.15675121	12.48289866
1423848231	411660	3	30.54544731	11.10033003
1423848231	1677109	4	30.5523774	13.15698712
1423848231	1586458	5	29.26831936	9.733483534
1423848231	1587070	5	28.84411312	9.734913051
1423848231	1666516	6	30.54775729	11.78585244
1423848231	1722389	7	30.97427376	12.47003532
1423848231	1694132	8	26.30371852	11.11462342
1423848231	1694345	8	25.87962348	11.11605249
1423848231	1702414	9	28.42925873	12.47861088
1423848231	1710890	10	27.14982096	10.42624237
1423848231	1683094	11	27.57622604	11.11033544
1423848231	1683528	11	27.152131	11.11176452
1423848231	1721392	12	30.96965355	11.09890058
1423848232	1775571	1	27.58084626	12.48146965
1423848232	1691981	2	27.15675121	12.48289866
1423848232	412647	3	30.54544731	11.10033003

3. How can players be tracked from video data, using combinations of computer vision, data association and control theory techniques?

The biggest challenge is to cope with the extensive camera changes that occur throughout the videos. Due to this non-static behaviour and the inherent properties of cameras, there are frames which are blurry and do not correctly display the 3D world as is. Secondly, the players are filmed under a relative small angle with respect to the horizon. This causes occlusions to occur more frequently compared to a larger angle. This makes detections of players harder, if not impossible when full occlusions occur. More importantly, due to the small angle, small deviations in detection-locations in the vertical component, can have drastic consequences for its back projection onto the 2D ground plane. This phenomenon is explained in Section 4.3.5. Also, the surroundings are viewable throughout the entire video sequences, which includes the crowd and/or audience. This will affect the detection accuracy negatively as well.

Because of the camera setup - on a tripod, close to the field, small angle with respect to the horizon, regular zoom-ins on actions - not all players will be visible in each and every frame. In such cases players cannot be detected and their location has to be estimated. Lastly, and this extends the second challenge: because there is audience visible on the frames as well, these people might become part of a players' track. Incorrect detection identification can result in non-player detections being recognised as player detections, and will therefore be used as input for the tracking algorithm.

2.4 Project relevance

This project finds its relevance directly in the gained value of the overall insights that the Dutch team will have. Having location data of both teams allows data analysts to perform more detailed analyses on the matched that are played. In the two sections below the relevance to the world in general, and to the scientific community are stated.

2.4.1 General

Not only is this project interesting for the Dutch rugby team, but its processes and models are widely applicable to any rugby team that has similar data (video data and GPS). In fact, the processes are not bound to rugby, or any sports, but can be used with any object of known appearance. Where GPS information is normally rather noisy, and only performs measurements at a low frequency, the combination with video data enriches the location data and makes it more accurate as well.

When only considering the video analysis, it is noticeable that a solution to our problem results in a very low cost opportunity for sports team to get analysis data. Because video cameras are relatively cheap compared to a set of GPS receivers, this provides a low cost alternative to this *traditional* way of obtaining location data. Additionally, video data also contains information about the opponent, which is not included in a regular GPS data set. This allows data analysts to perform more detailed analysis on the interactions between both team members as opponents.

2.4.2 Image processing and computer science

Object detection and tracking are hot topics within the scientific community, with dozens of papers being published per year. This includes the creation of new methods as well as the combination of existing methods. The combination of processing video data and GPS data at the same time is a unique one. To the best of our knowledge, there exists no other research in which object GPS data was combined with video data of that object. This makes this part of our research question a very interesting one, and will allow for interesting discoveries.

Our focus will lie within the geometric modelling of the camera pose. This brings some additional insights in how the frames can be modelled from frame to frame. For camera pose estimation, much research has been performed on deriving relative camera poses between individual, non-temporal related images. However, not much research is available that deals with continuous pose estimation in video, which has more stringent constraints due to the temporal relations between the frames. The problem at hand deals with a static location for the camera, which makes it possible to define relative poses from only a couple of frames. This concept will be introduced in Section 4.3.3.

Chapter 3

Background and related work

In this chapter we discuss the most relevant techniques for our challenge. We start with the relation between the 3D world and the 2D image, and how a pixel on an image relates back to a series of coordinates in the 3D world. Then, we discuss person detection from images and video. Lastly, we explain concepts and methodologies of tracking, which allows us to couple the detections of single frame into a whole, such that a string of consecutive detections is created.

3.1 Camera modelling: a pinhole camera

Cameras have the ability to create a 2D image from a 3D world. This ability can be modelled by assuming the camera follows a pinhole model. In this model, the light is modelled as rays and the lens is considered a small hole. Although current day cameras consist of a series of lenses in order to allow for multiple focal points and to focus light effectively, it is assumed that the combination of all the lenses can be modelled as a single *super*lens and hence as a small hole. The image plane is at the opposite side of the camera's lens and shows an upside down image. Therefore, the virtual image is used in front of the camera's lens, which is more convenient. In Figure 3.1 a schematic of the camera is shown on which the camera model is based.

In terms of equations, the camera model that projects world coordinates onto the virtual

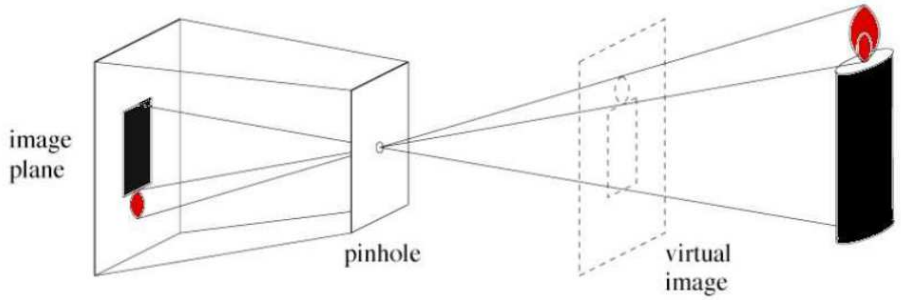

Figure 3.1: A schematic of the pinhole camera

image plane through a perfect pinhole camera looks as follows:

$$\begin{aligned}
\lambda \mathbf{x}^{im} &= \mathbf{P}\mathbf{X}^{W} \\
\lambda \mathbf{x}^{im} &= \mathbf{K}[\mathbf{R} \mid \mathbf{t}\ ]\mathbf{X}^{W} \\
\lambda \mathbf{x}^{im} &= \mathbf{K}[\mathbf{R} \mid -\mathbf{R}\mathbf{C}\ ]\mathbf{X}^{W} \\
\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} &= \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{bmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix}
\end{aligned} \tag{3.1}$$

In this model, a homogeneous world coordinate $\mathbf{X}^W$ is projected onto homogeneous image coordinate $\mathbf{x}^{im}$, using a 3×4 camera matrix P. In this model λ represents a homogeneous scaling factor. It represents the distance of the world point to the camera center. Because an image has no depth, all intermediate points on the line from the world coordinate to the camera center, are projected on the same image coordinate.

The camera matrix $\mathbf{P}$ consist of two parts. $[\mathbf{R} \mid \mathbf{t}]$ is called the extrinsic matrix. It transforms the 3D world coordinates into 3D camera coordinates following a rotation ($\mathbf{R}$) and a translation ($\mathbf{t}$). The $\mathbf{K}$ matrix is called the intrinsic matrix, and transform the 3D camera coordinates into 2D image coordinates.

The axes for the image plane F_I are defined as follows (see Figure 3.2).

- The origin of the image plane is at the left top of the image.
- X_I is the horizontal axis, increasing towards the right.
- Y_I is the vertical axis, increasing from top to bottom.

3.1.1 Dealing with distortion

The model assumes that lines that are straight in the real world, are straight on the image as well. However due to several distortion factors, this is not the case with most modern day cameras. These distortion factors have geometrical influences. These distortions encompass, among others:

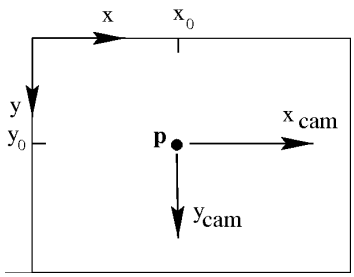

Figure 3.2: The image plane coordinate system. The origin is put on the left top of the image plane. Point **p** refers to the principal point of the camera, which is the point on the image plane that intersects with the z-axis Z_C of the camera coordinate system. Adapted from [13].

- Radial distortion, a geometrical effect which causes a barrel or cushion-like representation. Typical (but extreme) examples are the images produced with fish-eye lenses. It originates from the symmetry of the camera's lens system, in which light rays are bend by the camera's lens. The effect degrades with larger focal lengths.
- Tangential distortion, a geometrical effect that causes a non-symmetrical effect similar to radial distortion. This is caused by misalignment of the camera's lenses.

For an extensive explanation of geometrical and radiometrical distortion, please refer to [27, 25].

Fortunately, it is not difficult to correct for radial and tangential distortion, which are the main geometrical distortions. They cause the largest part of the non-linearity. The correction performed in radial direction removes the so-called fish-eye or barrel effect. It is defined by three radial distortion coefficients, κ_1 to κ_3

$$x_{corr}^{im} = x(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \tag{3.2}$$
$$y_{corr}^{im} = y(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \tag{3.3}$$

Here, r is the distance from the Euclidean distance from the principal point of the image plane. In the tangential direction the the correction corrects for the camera lenses being not entirely parallel to one another. These are defined by p_1 and p_2.

$$x_{corr}^{im} = x + \left[2p_1 xy + p_2(r^2 + 2x^2)\right] \tag{3.4}$$
$$y_{corr}^{im} = y + \left[p_1(r^2 + 2x^2) + 2p_2 xy\right] \tag{3.5}$$

3.1.2 Deriving the extrinsic matrix

The extrinsic matrix maps points which are defined in a global (or world) coordinate system F_W to points in the coordinate system of the camera F_C. In order to find the transformation matrix, two parts have to be identified. The rotation of the world coordinate system with respect to the camera coordinate system. And secondly $\mathbf{t}$, the position of the world origin O_w with respect to the camera, expressed in F_C. This relation is shown in Figure 3.3, and is expressed as follows:

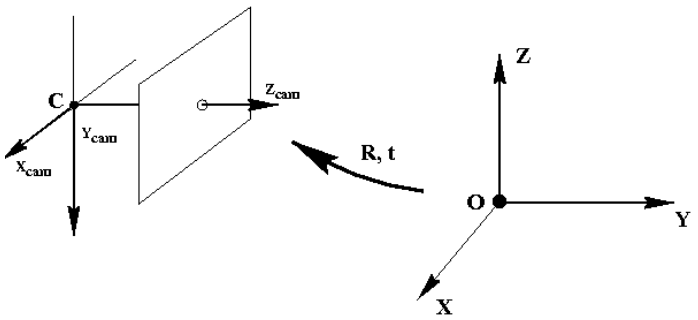

Figure 3.3: The relation between the world coordinate system F_W and the camera coordinate system F_C, given be $\mathbf{R}$ and $\mathbf{t}$. Adapted from [13].

$$\begin{aligned} \mathbf{X}^C &= [\mathbf{R} \mid \mathbf{t}]\, \mathbf{X}^W \\ \hat{\mathbf{X}}^C &= \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \mathbf{X}^W \\ &= \begin{bmatrix} \mathbf{I}_3 & \mathbf{t} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{R} & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} \mathbf{X}^W \end{aligned} \tag{3.6}$$

Here, $\mathbf{X^C}$ refers to the non-homogeneous coordinates in F_C, and $\hat{\mathbf{X}}^C$ to the homogeneous ones. In this model the axes for the coordinate system F_C are defined as follows:

- The origin of F_C is at the center camera's lens, i.e. at the pinhole.
- Z_C is the 'look-along' axis, and it is the axis through the principal point (p_x, p_y). A rotation around this axis is called roll.
- Y_C is pointed downwards. A rotation around this axis is called pan.
- X_C completes the right handed coordinate system. This means its goes towards the left (seen from behind). This is in correspondence with the described image plane above, and is shown in Figure 3.3. Rotations around X_C are denoted tilt.

Since it is more natural to express the location of our camera with respect to the world coordinate system F_W, the system can be rewritten to incorporate that position. This is especially convenient for the problem at hand. Since the camera's location is fixed, it can be described by a static parameter $\mathbf{C}$ depicting the position of the camera in F_W. Note that this is different from the variable $\mathbf{t}$, which is the position of the origin of the world O_W expressed in F_C. Rearranging Equation 3.6 such that $\mathbf{C}$ is used instead of $\mathbf{t}$ gives us

$$\begin{aligned} \hat{\mathbf{X}}^C &= \begin{bmatrix} \mathbf{R} & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I_3} & -\mathbf{C} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \mathbf{X}^W \\ &= \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \end{aligned} \tag{3.7}$$

Or simply $\mathbf{X}^C = [\mathbf{R} \mid -\mathbf{RC}]\mathbf{X}^W$. From this inversion, we can derive the relation between the original and more intuitively defined translation vector $\mathbf{t}$ and $\mathbf{C}$ respectively:

$$\mathbf{t} = -\mathbf{RC} \tag{3.8}$$

3.1.3 Deriving the intrinsics

Given the object coordinates in F_C, the intrinsics allow the coordinates to be mapped onto the image plane, as shown in Figure 3.4. Within the pinhole camera model, a point $\mathbf{X}$ is mapped onto image point $\mathbf{x}^{im}$. It is the point on the line from the camera centre to $\mathbf{X}$, where it crosses the image plane. The relationship is shown in 2D on the right part of Figure 3.4. Every point on the line from O_C to $\mathbf{X}$ will therefore be mapped onto the same point on the image plane.

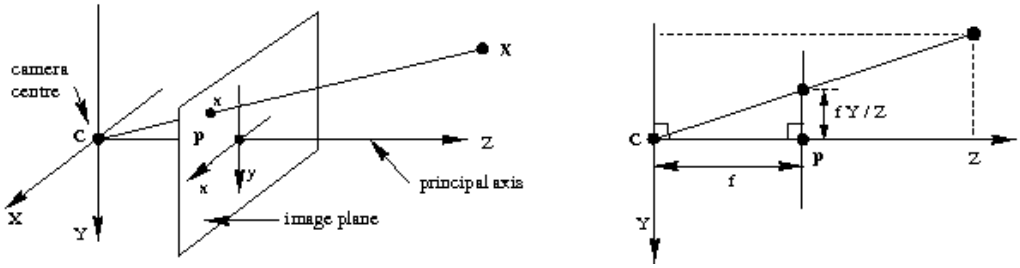

Figure 3.4: The relation between camera and image coordinates, given by intrinsic matrix $\mathbf{K}$. A point $\mathbf{X}$ is mapped to the image plane such that lies on the line from $\mathbf{X}$ to the camera centre, and intersects with the image plane. Adapted from [13].

This relation between coordinates in F_C is similar in the x-direction. So, our point $\mathbf{X} = (X, Y, Z)^T$ is mapped onto $(fX/Z, fY/Z, f)^T$. If the last element is excluded, we see that there is a Euclidean mapping from $\mathbb{R}^3$ to $\mathbb{R}^2$.

The image plane however, has its origin defined at top left corner of the picture, whereas our current mapping assumes it is at the intersection with the principal axis of O_C. The image plane coordinates found are therefore translated with the distance from the principal axis (p_x, p_y). In matrix form this results in

$$\lambda \mathbf{x}^{im} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{X}^C \tag{3.9}$$

Generally, a skew factor s is added to the (1,2) entry of the K matrix, which is defined as $s = \tan(\Phi)$, with Φ the angle between the image axes. This refers to the image pixels not being rectangular, but are seen as parallelograms. Since there is no skew in many modern day cameras, as well as the video data for this thesis, it is left out for convenience.

Note that the intrinsics are closely related to the distortion coefficients. $\mathbf{K}$ assumes a perfect mapping from camera coordinates to image coordinates, in which 3D coordinates are mapped to 2D coordinates, along straight lines. The distortion coefficients correct for the practical offset, errors and inconsistencies of this mapping. When we combine the information from both the intrinsics and the extrinsic, we can see that we have 4 parameters to define our intrinsic matrix $\mathbf{K}$, 5 for the distortion vector $\mathbf{d}$, 3 parameters for rotation matrix $\mathbf{R}$ and 3 for translation vector $\mathbf{t}$. This gives our pinhole camera model a total of 14 parameters.

The intrinsics of a camera are independent of the view seen by the camera, and is constant for changes in rotation and translation. Adjustments in focus or zoom level do change the intrinsics. Since the distortions as described above are dependent on the intrinsics of the camera only, these will change with the intrinsics accordingly. To find the intrinsics, a camera calibration needs to be performed.

3.1.4 Camera calibration

The methods available today for camera calibration can be divided into two categories; photogrammatic calibration, and self-calibration. Photogrammatic calibration makes uses of a calibration object, of which its geometry in 3D is known up to a high precision. Often a planar object is used, such as a chess-board. The most common and well known method used is described by Zhang [38], which allows for calibration with as few as only 4 different calibration images. It is also able to calculate the geometrical distortion up to 8 degrees. This methods requires a series of images of the calibration object taken from different angles. It uses the 2D-3D point correspondences from the different images to derive the homography between them. The homography matrix is a 3×3 matrix, that performs a mapping from $\mathcal{R}^2$ to $\mathcal{R}^2$, giving the affine transformation of the object plane in the first frame to the second frame up to a scale level such that $s\mathbf{x}' = \mathbf{H}\mathbf{x}$. The homography also relates the intrinsics of the two frames, through the absolute conic of the image:

$$\omega = \mathbf{H}^{-T}\omega\mathbf{H}^{-1} \tag{3.10}$$

Here $\omega = \mathbf{K}^{-T}\mathbf{K}^{-1}$. The homography is defined by 8 parameters, whereas ω is defined by 6, meaning that for each set of point correspondences, two parameters can be estimated. Having more than 3 images allows one to derive the initial camera matrix $\mathbf{K}$, by solving it for a least square solution.

The extrinsic are found by initial pose estimation of each frame with a perspective-n-point (PnP) solver. The solver finds a solution to the extrinsics of each of the images, given the 2D-3D point correspondences and the intrinsics. The camera parameters are updated during a final optimisation step, in which a Levenberg-Marquardt algorithm tries to minimise the reprojection error of the image points. In this step the estimation of the distortion coefficients is also included. The optimisation step tries to optimise

$$\sum_{i}^{n}\sum_{j}^{m} ||\mathbf{x}_{ij} - M(\mathbf{K}, \mathbf{D}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_{ij}||^2 \tag{3.11}$$

where M is projection function that maps the 3D world coordinate $\mathbf{X}_{ij}$ onto the image frame, using the intrinsics, distortion vector and the extrinsics.

For zoom lenses, this process is repeated at different zoom levels, all for which a series of calibration images is present. Finally, the parameter values in between measurements are found by interpolating between the surrounding data. Methods have been found that interpolate linearly, exponentially or who use third degree spline fitting. The zoom range process is found in [33, 37] and [28] to name a few.

On the other hand, there is self-calibration, which is the calibration of a camera, by simply pointing it at a rigid scene. The scene needs to be very rigid, not allowing for wind or light intensity changes, and the actual size of objects in the world need to be estimated. In comparison with calibration with a known object, auto calibration is a more difficult method. By taking multiple shots of the scene, from different perspective, but with the same zoom level, the relative orientation and position of the scene can be determined, in a manner similar to photogrammatic calibration. Here, feature points - interesting, distinctive points which are small rich textured area in which there are intersecting edges - are used as reference points between the frames. The points are chosen by a feature detection algorithm such as Harris corner detection or the Scaled Invariant Feature Transform (SIFT) (see Section 3.2.2 for more on feature detection). These features are then matched through the different frames, and the homographies are derived. For a sequence of images, the intrinsics are kept constant and the homographies can then be used to solve for the intrinsics using the absolute conic of the image, similarly to the initialisation of the photogrammatic calibration. Note that this does not include any distortion correction.

Sturm [26] was one of the first to introduce the inclusion of radial distortion for a zoomable camera, by performing a series of *pre-calibrations* before the self-calibration, such that the intrinsics could be estimated with a single parameter. Sinha & Pollefeys [24] propose a method for zoomable cameras. They start with a single self-calibration at the minimum zoom level. Then they perform a zoom sequence to capture the differences between different zoom levels and derive all intrinsic parameters from this sequence.

3.1.5 Camera motion tracking

When determining the ground plane location data of players, it is essential to know the camera parameters at every iteration. Therefore, the camera motion needs to be tracked as well. Roughly speaking, there are three ways to determine the motion of the camera.

First of all, there is background feature tracking, a method in which a series of keypoints are selected and matched in both frames, between which the homography is calculated. From this, given that the intrinsics of one of the frames are known, the relative rotation between the frames is found, as well as the new intrinsics. In terms of equations, by multiplying the found homography by the old intrinsics, we get:

$$\begin{aligned} \mathbf{H}\mathbf{K}_1 &= \mathbf{K}_2\mathbf{R}_2\mathbf{R}_1^{-1}\mathbf{K}_1^{-1}\mathbf{K}_1 \\ &= \mathbf{K}_2\mathbf{R}_{12} \end{aligned} \tag{3.12}$$

which can be solved for $\mathbf{K}_2$ and $\mathbf{R}_{12}$ using an RQ decomposition. With a RQ decomposition a matrix ($\mathbf{A}$) is split into two matrices, an orthonormal matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$, such that $\mathbf{A} = \mathbf{R}\mathbf{Q}$. $\mathbf{Q}$ is found by finding an orthonormal basis for Col$\mathbf{A}$ = Span$\{x_1, x_2, x_3\}$, using the Gram-Schmidt process [16]. To complete the RQ decomposition, $\mathbf{R}$ is found from

$$\mathbf{A}\mathbf{Q}^T = (\mathbf{R}\mathbf{Q})\mathbf{Q}^T = \mathbf{R}\mathbf{I} = \mathbf{R} \tag{3.13}$$

The second method is one where a marker is used, similar to one that is used during photogrammatic calibration: An easily recognisable object, of which the location and size

is known in world coordinates. A PnP solver finds the extrinsics $[\mathbf{R} \mid \mathbf{t}]$. Kukulova [4] has introduced a method that is able to incorporate the intrinsics and distortion variables as well. Methods such as Taketomi et. al [28], do a zoom level guess to use the intrinsics from pre-calibrated data. They then use a PnP solver to find the rotation and translation of the camera. They iteratively solve for the correct zoom level, by minimizing the reprojection error from a marker and the epipolar constraints as well as a zoom continuation.

Lastly, there is the method of optical flow tracking. In optical flow, an estimation is made on the trajectory of points from time-varying image intensities. The process tries to determine the position of a small patch of an image on a target image. In video, typically, pixel locations change very gradually, which limits the search space of previously selected pixels. Optical flow is caused by object motion, camera motion or light intensity changes. Optical flow assumes that the intensity of pixels undergoing motion do not change intensity and it searches for for nearby pixels with a minimum difference in intensity.

A very common optical flow tracker is the Kanade Lucas Thomassi (KLT) tracker [27], which builds on the Lucas-Kanade method. The KLT tracker can be used to track background features, or foreground features. Based on an initial set of selected features, by any of the feature detection methods as described in Section 3.2, the algorithm searches for these points in the next frame. In the first frame, for which the camera parameters are assumed to be known, a projection of the points can be done when one of the coordinates of the point is known. In case of ground plane points, the Z^W coordinate is always 0. For each of the tracked image points a 2D-3D correspondence is then available in the subsequent frames, and with methods similar to the ones described above, the camera parameters can be derived.

3.2 Object detection

Finding objects within images is a key ability for a large variety of computer vision problems and real life applications. Object detection is a technology that tries to localise an object, or a class of objects of known size and appearance within an image. We discuss the people and pedestrian detection in this section.

According to the elaborate survey by Dollar [9], the most promising pedestrian detectors up to date use at least one of the following ways to describe the object and the (partial) image: gradient histogram, gradients/edges, pixel intensities, colour, texture, self-similarity or motion. Typical object detectors range from using a single feature type to using a number of them. Typical single feature type detectors are SIFT [19], Histogram of Oriented Gradients (HOG) [6], the Viola-Jones object-detection framework [30] using Haar-like features and the 'edgelet' feature detector introduced by Wu and Nevatia [34], which all focus on edges and gradients. This is because it is the most light intensity and colour invariant feature descriptor.

Multifeature detectors include the detector of Wojek and Schiele [9], which use a combination of Haar-like features, shapelets, shape context and HOG. The authors show that it outperforms the use of a single feature. Another detector in proposed by Wang et al. [31] with their combination of HOG and local binary patterns, and the "fastest pedestrian detector in the world" [8] which practically combines every feature type but motion, and is

able to approximate features for a range of scales when only computed at a single scale. The performance of the multi feature detectors are generally better than the single feature ones on terms of miss rate and false detections.

A new method of object detection is to use a part based model detector. These models make of the pictorial structures idea, where an object is split into multiple parts. These parts independently detect specific parts of an object(e.g. wheels on a car, limbs on a person, steers on bikes, faces). There is an accompanying model on the topology of the object, stating relative positions of the different parts, along with a deviation policy. This add a geometrical feature to the detection process, which has not been taken into account in the above discussion. Part-based models have the advantage that objects are no longer restricted to a single pose, but are allowed to vary in size, shape and articulation. This makes them especially interesting for detection within sport scenes.

An important part of the part-based model is the pictorial structures idea, where an object is represented as the joint configuration of its parts. The joint configuration is defined by the topological model and describes the relative position of the different parts. This describes the mean distance as well the variance of the location of parts with respect to one another. An important component of this approach is to model the *a priori* knowledge on the possible object configurations. Note that such a model assumes that an object is always viewed in a manner similar to the training data, which is most commonly restricted to a single view. Due to the deformability of parts rotations and deformations are possible, but only up to some small extent.

3.2.1 Felzenszwalb: Object detection with a part based model

One of the most well-understood and successful approached for general object detection to date is the approach of Felzenszwalb et al. [11], which uses a part-based model, and is based on the HOG descriptions of images. In 2006, Dalal and Triggs published their methodology of detecting humans by using histograms of oriented gradients [6]. In their paper they proposed a descriptor that was based on the distribution of local intensity (edge) gradients. The HOG descriptor has better properties than simple edge detection, since its description makes it invariant to illumination and local variations. Furthermore, it is invariant to rotation up to some small degree.

The framework of Felzenszwalb et al. [11] is built to detect an object as a whole by the so-called root filter, as well as individual parts of an object by part filters. For each of the filters there is a trained model, that describes the associated part by a feature map. This feature map is defined by the HOG model. The different parts are related to one another according to the pictorial structures model, in which parts connections are presented by strings. The model specifies the anchor position of each part with respect to a single or multiple reference parts. The deviation of the part from this anchor position is seen as stress on the connecting string, and gives a deformability cost. Felzenszwalb et al. give their parts anchor position with respect to the root filter only, to reduce complexity.

Detection is initiated by creating an image and feature pyramid that describes the image at different scales, in order to find the objects of different sizes. The feature pyramid is the HOG-described equivalent of the image pyramid. Next, at each of the different layers in the feature pyramid, a root filter response is calculated, based on the similarity between

the filter and the subwindow. Similarly, responses are calculated for the parts. Since the model is based on the deformability of parts, the part responses are transformed to allow for spatial uncertainty. That is, parts are allowed to be positioned away from the anchor position as described in the model, but it will come with a deformation cost. Finally, the response of the root filter is combined with the transformed responses of the parts, and this leads to a final response or score for an object. The detection and matching process is shown in Figure 3.5.

The procedure as described above is performed for different components, which all have their own model. Each component might refer to a different view of the object (side-view, front view, top view) or to a different pose (standing, lying, running) for example. Each model is given its own bias to add to the score of the model. The final score for a detection is determined by the highest scoring component. The components together create a mixture model $M = M_1, M_2, ..., M_m$ where M_c refers to the model of component c.

An extension of the Felzenszwalb framework, as described in [10], shows that part detections and transformations are only applied if the total score of the root location and the parts detected up to the current part is above a certain threshold. This avoid having to evaluate the image at locations on which it has already been concluded that there is a very low probability of an object's presence. Figure 3.6 shows an example of evaluated locations on an image.

3.2.2 Feature detection & matching

Feature detection is the process of describing an image by a series of local image features. Collectively, a set of features describe an object. The features of most interest are found in high contrast regions, with intersection edges. These regions contain the most distinctive information. The features have a descriptor which is ideally invariant to translation, rotation, scale, illumination, 3D and affine projection.

The set of keypoint descriptors of two images can be used to find similarities between two images. By comparison of the feature descriptors, and by matching those who are most similar, local areas of the first image are related to local areas of the second one. There are many applications for a such a set of matched keypoints. Object recognition and matching, 3D scene reconstruction and motion tracking, these all rely on the presence of stable, representative features in the image. The ideal keypoint descriptor describes the most prominent and distinctive information of an image regions, such that if the image regions is present in another image, the points can be matched.

Although the identification of local interest points can be traced back to the Harris corner detection [12], one of the first approaches for robust feature description and matching was described by Lowe [19]. In his paper on scale invariant feature transform (SIFT) Lowe proposed a 4-step algorithm to find and describe keypoints of an image.

- Scale space extrema detections
- Keypoint localisation
- Orientation assignment
- Keypoint description

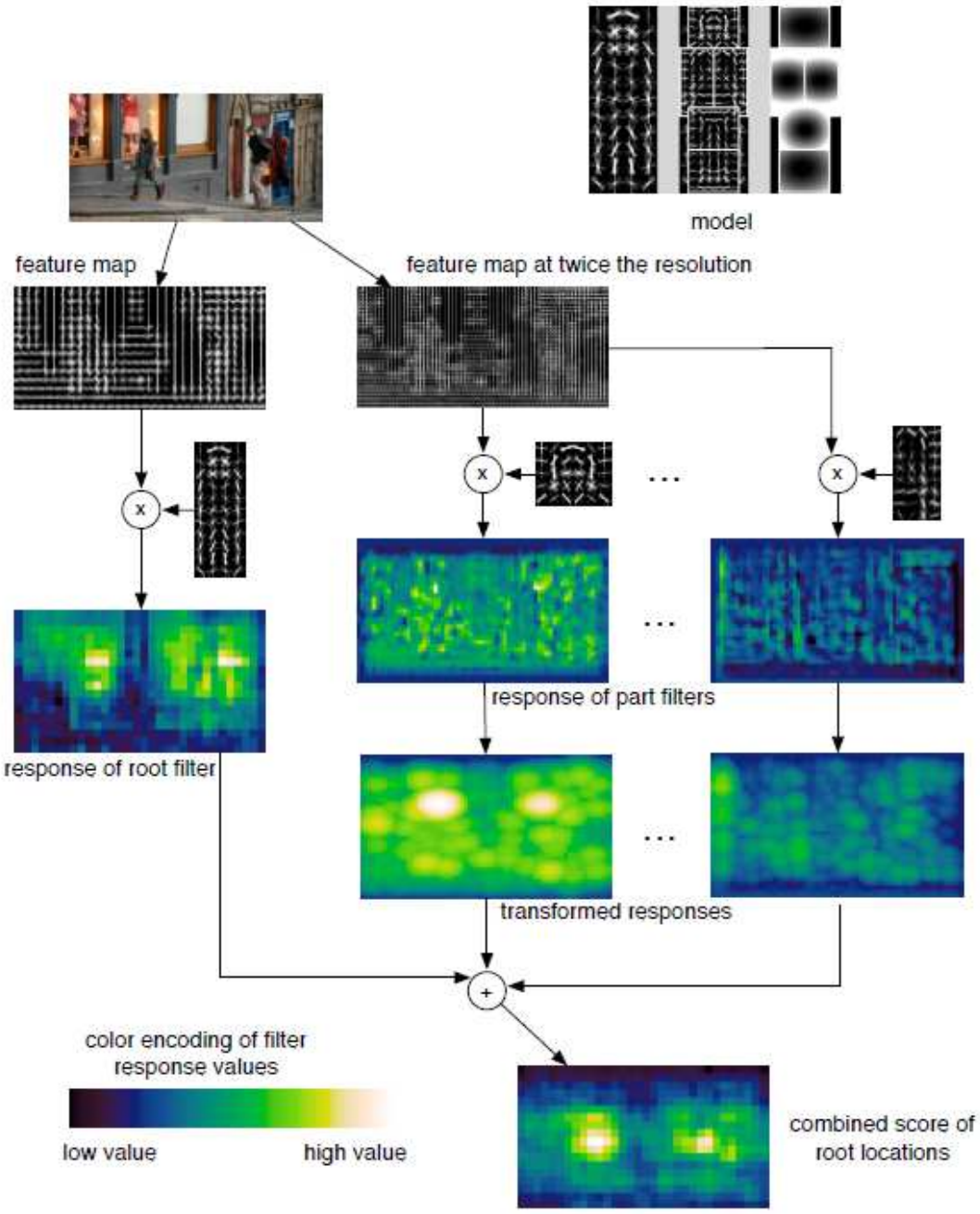

Figure 3.5: An overview of the detection and matching process within the Felzenszwalb framework. Responses from the part filters are computed from the image at twice the resolution. The responses of the part filters are then transformed to compensate for spatial uncertainty. Finally, the responses of the parts and the root filter are combined to give the final probability distribution. Here, the head part and the right shoulder are given as two example parts [11].

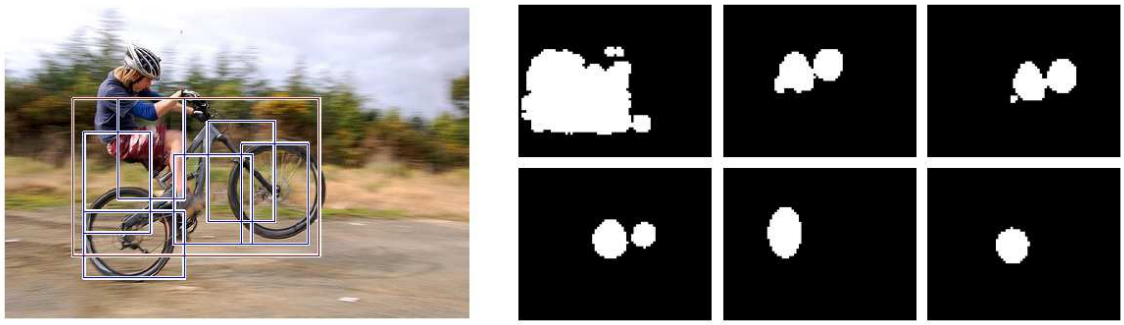

Figure 3.6: An illustration of the cascade object detection framework of Felzenszwalb. Each image on the right shows the evaluated locations for a part. The cascade order was performed from left to right, top to bottom. It clearly shows that the last few parts are hardly evaluated because of the previous results eliminating the other positions. [10].

His keypoint descriptor describes each keypoint by a weighted sum of edge gradients of the surrounding pixel. The first step is to select scale-space extrema. The target image is used to create an image pyramid, in which copies of the image are resized to different octaves, where each subsequent octave has half the image size of the previous octave. At each octave, a Gaussian filter is applied such that a series of images with different levels of filtering result, called scales. (see Figure 3.7). For two adjacent scales the images are subtracted, and this gives a difference of Gaussian (DoG). For each image and the two adjacent ones in the DoG pyramid, the minimum and maximum differences in a $n \times n$ neighbourhood is saved (typically 3x3).

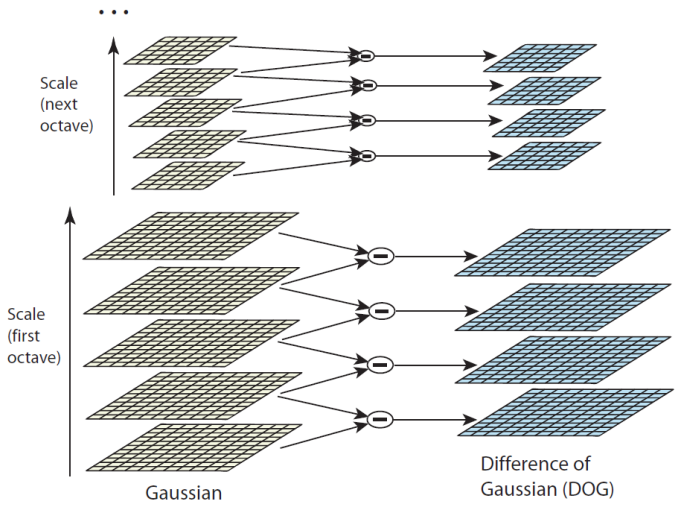

Figure 3.7: The image pyramid used to determine the scale space extrema. At each octave, adjacent scales (levels of Gaussian filtering) are subtracted. The resulting Gaussian-of-difference image pyramid are shown on the right [20].

In step two, keypoints along an edge, or with a low contrast are removed from the solution set. Of the remaining keypoints, the edge gradient is set as the orientation of the keypoint.

To create the final keypoint descriptor of step 4, the location and the scale used to find the keypoint are used together with the orientation. The pixels from the area around the keypoint form the basis of the descriptor.

The set of 16×16 pixels around the keypoint location is selected, and their gradients are rotated relative to the keypoints' orientation. Then, each pixel is weighted according to a Gaussian weighting function with σ equal to half the size of the descriptor window, in order to give higher importance to pixels close to the keypoint. For each 4×4 subregion, a histogram of 8 orientation bins is created. The weighted edge strength of the pixels in the subregion are added to the bin corresponding to the edge gradient and each subregion is then described by a single gradient histogram. This is shown Figure 3.8. The resulting vector is found by concatenating the normalised subregion vectors.

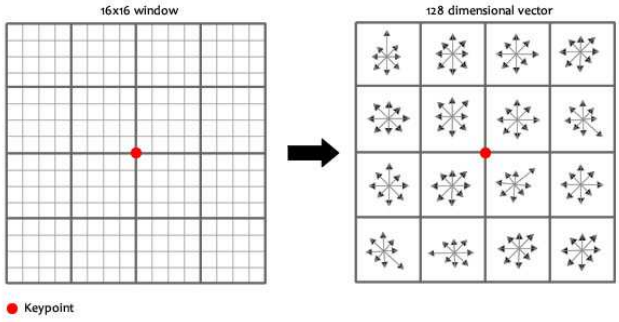

Figure 3.8: The SIFT keypoint descriptor is described by a series of histograms. In each 4×4 subregion a gradient histogram is build by accumulating the edge strengths of the pixels within the region. These strengths are Gaussian weighted, giving higher importance to pixels closer to the keypoint.

Since SIFT has been introduced many new and updated feature descriptors have been proposed. A short explanation of the most prominent ones is given below.

SURF [1] The speeded up robust feature detector (SURF) shows an improvement on all parts. The feature selection is improved by an improved version of corner detection via an approximation of the Harris matrix, done by a box filter. This allows a significant speed up for detection, reported to be more than 5 times faster than the DoG from SIFT. It also provides a small extension which provide even more speed, called upright SURF. This extension which is only invariant to approximately 15° rotation.

ORB The method Oriented Fast and Rotated Brief, gave a good alternative for real-time applications [17]. It combines a FAST keypoint detector and a BRIEF descriptor. FAST looks at intensity differences between a selected pixel's neighbourhood top and bottom pixel, and uses this to classify whether the neighbourhood is a corner. This method is fast, but uses a fixed pixel neighbourhood and is therefore not scale invariant. BRIEF is a descriptor that describes the keypoint's neighbourhood as a binary string. As a result, keypoints can be compared by using the Hamming distance, which is more efficient than the commonly used L2-norm. Due to this construction, BRIEF is not rotation invariant.

ORB improves upon FAST and BRIEF in threefold. Firstly, it adds an orientation component to the FAST keypoint detection algorithm, as well as an image pyramid

feature, such that it becomes both scale and rotation invariant. BRIEF is improved by using the orientation component, and describing the keypoint relative to its orientation. This makes ORB more invariant than SIFT or SURF. It is also an order of magnitude faster than SURF, and two orders of magnitude faster than SIFT. This makes ORB a good candidate as real-time keypoint descriptor.

BRISK This method uses a FAST-based detector. It extends FAST to make it invariant to scale by making use of an image pyramid in which there are 2 images per octave. It assigns an interpolated scale to a keypoint, instead of picking the scale with best result.

The descriptor is based on BRIEF, with some minor adaptations. There is a derotation step, in which the descriptor is made invariant to rotation. This is done by intensity comparisons of the image patch, which is Gaussian smoothed to prevent aliasing effects. To compare the descriptor to other descriptors, the Hamming distance can be used. When compared to ORB, the performance in terms of repeatability and complexity are similar. The main difference comes from the derivation of the keypoint orientation.

3.2.3 Colour based image descriptor

One other approach is to look for an object on the basis of colour. For objects with a single colour, or series of colours, a colour model can be created specifying the appearance of the object. Alternatively, colour can be used to determine whether an object which is found using any other method complies with the colour prerequisites of that object. For example, if one is looking for people wearing an orange outfit, the object detector should be able to distinguish between colour.

A colour model can be based on, amongst others, a mean colour, a Gaussian mixture model, or a colour histogram. A mean colour is a single colour, described in any colour space, and is used to indicate the most prominent colour of an object. Most often a range of colours is used to allow for robustness with respect to illumination.

To allow for many different colours, a colour histograms can be used. A histogram records the frequency of colours, where the colour range is made discrete into n bins. Colour histograms can be used for any colour space, and can either be made from single or multiple channel images. The RGB histogram is a combination of three 1D channel histograms, but is not invariant to any light intensity changes or shifts. The HSV histogram records the colour data in the Hue, Saturation, and Value colour space. It is common to not incorporate the V-channel in the histogram, since this refers to illuminations. The Hue channel is known to be unstable around the gray axis; when the saturation goes towards 0. Using Hue and Saturation is known to be invariant to light intensity and light shift changes [29].

A Gaussian mixture model (GMM) divides all the pixels of the reference image into a number of clusters, using a clustering method such as K-means. Each of the clusters is then described by a Gaussian distribution, and the resulting multi-modal distribution is found by adding the weighted cluster distributions (see Figure 3.9). A GMM is a compact way to describe an image representing an object that consist of small ranges of colours. The distance of a pixel in a target image can be efficiently calculated (e.g. by the Mahanalobis distance).

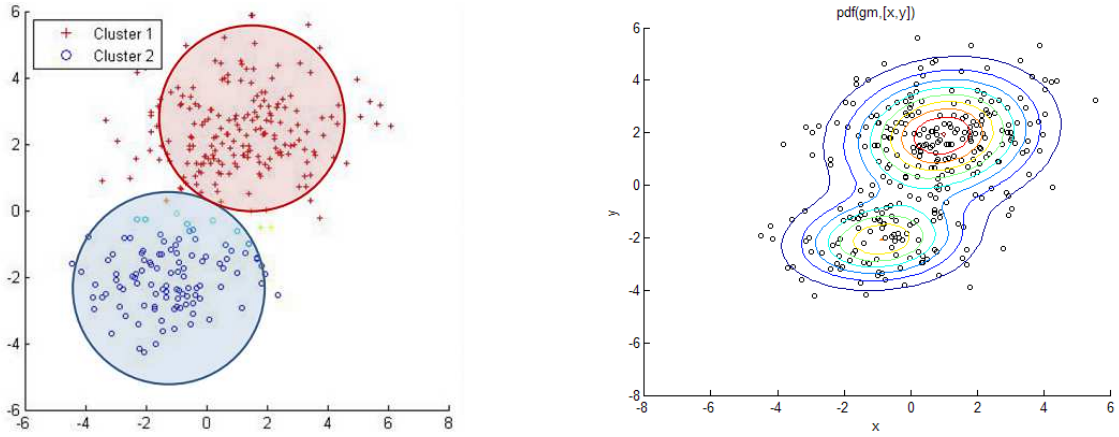

Figure 3.9: Creation of a Gaussian mixture model. On the left, clustering of pixel data is shown. For each cluster a multivariate Gaussian distribution is created. On the right is the resulting weighted sum of these distributions, showing the probability density function of the assesed sliding window [22].

3.3 Tracking

Video tracking is defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene [36]. Ground plane tracking is estimating the trajectory of an object relative to a (self) defined ground plane, such as the physical ground plane. By tracking , we refer here to ground plane object tracking unless stated otherwise, where objects are tracked with respect to a ground plane in the world. In our case this refers to the rugby field.

A brief literature review [36, 39, 35] reveals that there are many ways to track multiple objects. Globally these can be distinquished into two approaches. Tracking-by-detection, or detection-by-tracking. In tracking-by-detection, an image is extensively scanned by an object detector and the results are used as input for the tracking system where, based on several criteria, the detections are matched to the corresponding tracklets. A tracklet is a connected set of detections over time. Examples of tracking-by-detection approaches are online motion agreement tracking [35], in which pedestrians detections are matched to tracklets based on a cost function based on bounding box overlap, and a motion agreement argument. This argument compares the intent of the tracklet in terms of position and direction with the detections. Other examples are Wu & Nevatia [34], which use a part-based detector, and a matching function with cost based on location, size and appearance and Liu et. al [18], who perform an offline evaluation of all detections, by using a hierarchical trajectory association to build their tracklets.

With detection-by-tracking, the tracklets are initialised at the first frame, and at every next frame, a local search is performed in the neighbourhood of the previous detection in order to find the object in the next frame. It is only when an object is lost, that reinitialisation may take place by scanning the entire frame. An example of detection by tracking approaches is [7], where football players are tracked using particle filters.

In general, there are two types of criteria trackers use to associate a new detection to a tracklet: appearance and dynamics. The appearance is based on the similarity of previous

detections and a new one. Depending on the object representation, the appearance model can differ in terms object size, colour, rigidity, shape or contour. Typically, a person will not grow or shrink significantly over shorts periods of time (e.g. by coming closer), nor will it change its clothing, skin or hair colour. This is a tighter bound than the bounding box found by detection. The dynamics of a tracklet models and predicts the person's movement, i.e. motion agreement. A person will be inclined to move in the same direction as he did in the previous frame. Changes in both direction as well as magnitude will be graduate.

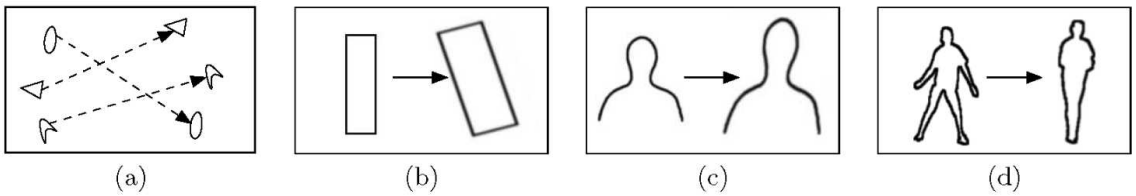

Figure 3.10: Tracking categories and approaches. (a) multiple point correspondences, point tracking. (b) parametric transformation, kernel tracking. (c) contour evolution, silhouette tracking.

Apart from the tracking approaches, there are three categories within object tracking, depending on the representation of the object: points tracking, kernel tracking, and silhouette tracking, see Figure 3.10. In point tracking, an object is represented by a single point in space, and association of points is based on the previously associated points. Only the dynamics of the points can be taken into account. Typical point trackers are the Kalman filter and particle filter for single object tracking, and the Joint Probability Data Association Filter and Multiple Hypothesis tracker (MHT) for multiple object tracking. A Kalman filter predicts the state of a linear system, often containing location and velocity, by assuming the states follows a dynamic model undergoing Gaussian noise. It corrects its prediction by information from measurements. The Probability Data Association filter is is a filter that only associates tracks with detections, based on the given model from the tracks. Each of the tracks can have its dynamic model defined in its own way. Multiple Hypothesis Tracking is a skeptic algorithm, which takes into account the detections over a series of frames, before assigning any of the detections to a track. This allows the MHT to deal with occlusion.

Kernel tracking is all about appearance. Typically, object positions are described by a bounding box, denoting the area in which the object is present. This allows trackers to incorporate appearance models to their track association parts. The track defines its object according to previously obtained detections, and tries to match it to the new detection. Typical examples are the KLT tracker as explained in Section 3.1.5, or the meanshift tracker. The meanshift tracker is an iterative algorithm that predicts the new location of the track, and based on the appearance of the predicted location, it iteratively updates the predicted location until the appearance similarity is maximised.

In the remainder of this section we describe the Kalman filter in more detail.

3.3.1 Kalman filter

The Kalman filter gives a recursive solution to the discrete-data linear filtering problem [32]. It is named after Rudolph E. Kalman, who published his paper in 1960 [15]. The Kalman filter is well known for its limited computational requirement, and elegant recursive properties, and is considered one of the most common data fusion algorithms today. The Kalman filter is typically used to smooth noisy measurement data and to provide good estimates of parameters of interest. The Kalman filter finds its origin in control theory, where state-space models are extensively used to create feedback and feedforward systems.

We discuss the discrete-time Kalman filter. Consider the following state space at time t that is assumed by the filter:

$$\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \mathbf{w}_t \tag{3.14}$$

The state variable $\hat{\mathbf{x}}$ contains the variables that are monitored, such as location coordinates or velocity, or detection size. The matrix $\mathbf{F}_t$ refers to the transition matrix, and gives the transformation from the old state to the new one. $\mathbf{u}_t$ is the control input. Matrix $\mathbf{B}_t$ is the control matrix, which transforms the control input to the state space. Together $\mathbf{B}_t\mathbf{u}_t$ state the changes that occur in the state space. Examples of uses of the control matrix comes from mechanical machines (e.g. cars, robots), where the actions of the car, such as pushing the gas pedal, or activating a servo is registered, and put in the control matrix. For person tracking, there is no control input. The last part of the equation refers to uncertainty. The vector $\mathbf{w}_t$ is drawn from the covariance matrix $\mathbf{Q}_t$, belonging to a zero mean, multivariate normal distribution ($\mathbf{w}_t \sim N(0, \mathbf{Q}_t)$).

Measurements can also be added to our state space, according to

$$\hat{\mathbf{z}}_t = \mathbf{H}\hat{\mathbf{x}}_t + \mathbf{v}_t \tag{3.15}$$

with $\mathbf{v}_t \sim N(0, \mathbf{R}_t)$, the measurement noise, coming from inaccurate or imprecise measurement devices. Matrix $\mathbf{H}_t$ gives the transformation from the state space to the measurement space.

The state of the Kalman filter is described by two variables, $\hat{\mathbf{x}}_{t|t}$, and $\mathbf{P}_{t|t}$, describing the a posteriori state estimate and covariance matrix at time t. The Kalman filter algorithm consists of two stages: a prediction stage, and an update stage. In the prediction stage the state in the next time step is predicted based on the dynamic model provided by the user, no measurement data is needed.

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t \tag{3.16}$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t-1|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{3.17}$$

The predicted variables are the *a priori* state estimate and covariance matrix respectively. The covariance matrix in Equation 3.17 is derived from the variance associated with the prediction of an unknown true value $\mathbf{x}_t$, and is given by:

$$\mathbf{P}_{t|t-1} = \mathbf{E}\big[(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})(\mathbf{x}_t - \hat{\mathbf{x}}_{t|t-1})^T\big] \tag{3.18}$$

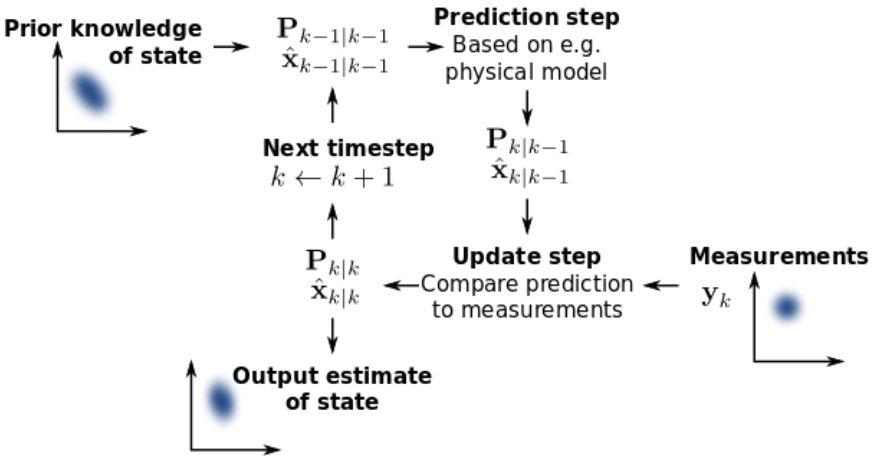

Figure 3.11: The basic workings of the Kalman filter. The real time linear state space estimator has a clear two step process: predict, and update, which are based on a dynamic model and measurements.

The measurement update is performed next.

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}) \tag{3.19}$$

$$\mathbf{P}_{t|t} = \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} \tag{3.20}$$

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{3.21}$$

The part $\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}$ is called the measurement innovation, and it reflects the difference between the prediction and the measurement. An innovation of null refers to the prediction and update being in complete agreement. The gain matrix $\mathbf{K}_t$ represents a blending factor that minimises the *a posteriori* covariance matrix. It sets a certain weight to the measurements, making them more or less trustworthy. Rewriting this matrix slightly results in a more intuitive expression:

$$\mathbf{K}_t = \frac{\mathbf{P}_{t|t-1}\mathbf{H}_t^T}{\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t} \tag{3.22}$$

Although the gain matrix can be expressed in many ways, the one stated above is the most common one, since it linearly scales the state towards the measurements. When looking at Equation 3.22 it can be seen that as the measurement covariance matrix $\mathbf{R}_t$ reduces, the measurements will be weighted more heavily, and the gain will approach the mapping from measurement space to state space. Similarly, when the *a priori* covariance $\mathbf{P}_{t|t-1}$ approaches zero, the measurements will have a lower weight, and the gain will drop to zero.

The state update is visualised in Figure 3.11. It shows the two-stage property of the Kalman filter. Since the prediction step is based on the previous state estimation, a prior state has to be provided by the user, as well as the associated covariance matrix.

Chapter 4

Methodology

In this chapter we present an overview of our methodology. We discuss the various assumptions and design choices, and present each step of our approach, following it chronologically.

4.1 Overall model

The goal of our problem, finding the ground plane location data of each player through time, can be translated into a state vector.

$$\mathbf{X}_t = (\mathbf{C}_t, \mathbf{R}_t, z_t, \mathbf{r}_t^1, \cdots, \mathbf{r}_t^n) \tag{4.1}$$

Here, t refers to the time, $\mathbf{C}_t$ is the camera position, $\mathbf{R}_t$ the camera orientation (tilt,pan,roll). The variable z is the zoom value of the camera and $\mathbf{r}_t^i$ is the ground plane position of player i. Since there are 14 players in a game of rugby sevens (with a 2 dimensional variable each), this leaves us with a problem of 35 dimensions.

4.1.1 Assumptions and dependencies

We simplify our model by taking the following assumptions.

1. The camera can be modelled accurately as a pinhole camera with radial and tangential distortion.
2. People stand on the ground.

3. The camera's position will not change over time.
4. The camera's roll parameter will not change over time.

As explained in Section 3.1, cameras are accurately modelled by the pinhole camera model throughout numerous computer vision applications. We therefore assume it as well. The second assumptions simplifies the reasoning and back projection of detections to the 3D world (F^W). By stating that people stand on the ground, the bounding box of the detection response on on the image is projected assuming that the lower edge of the bounding box is touching the ground.

Since the camera is placed on a tripod, the camera is not able to be moved around freely, in comparison with a handheld device. A quick look at our data confirms that the camera is never moved indeed, and so our second assumption will hold. Additional to the tripod restricting the movement of the camera, it also prevents the camera from rolling. Hence, the roll parameter is constant. The camera is only allowed to pan, tilt, and zoom. This reduces the state vector, since the camera's location only needs to be determined once. The same goes for the roll parameter.

Besides the assumptions, there are a number of dependencies between the state variables as well. The dependencies originate from the continuity of time and the laws of motion:

1. The pan and tilt angles will only change gradually from frame to frame. $\Delta\gamma$ (pan) and $\Delta\beta$ (tilt) are bound.
2. The zoom level of the camera will only change gradually from frame to frame. Δz is bound.
3. For every player; the movements are bound to the laws of physics and will therefore change gradually as well. Changes in location are therefore bound, and the velocities of players are bound as well.

Throughout the remainder of this chapter the effects of the assumptions and dependencies will be made clear.

4.1.2 Design choices

We have decided to go for a tracking-by-detection approach. Although the approach is generally slower than detection-by-tracking, it does allow us to split the problem into individual parts and have a larger distinction between the detection and the tracking part. Also, since the quality of the videos is not very high, we can easily derive information about the quality of the detections independently of the tracking part. To mitigate problems with occlusion, we use a Kalman filters for each of the tracklets - a string of associated detections through time, which allows a prediction of the players location and does not require a detection to be present at every frame.

By using tracking-by-detection, three main keypoints can be identified that coincide with the three sub questions defined in Section 2.3. First of all, there needs to be a model that describes the state of the camera: the intrinsics and extrinsics. The camera is allowed to zoom, and therefore the intrinsics are not constant. Calibration of the camera is explained in Section 4.2.3, and finding the camera parameters from previous frames is explained in

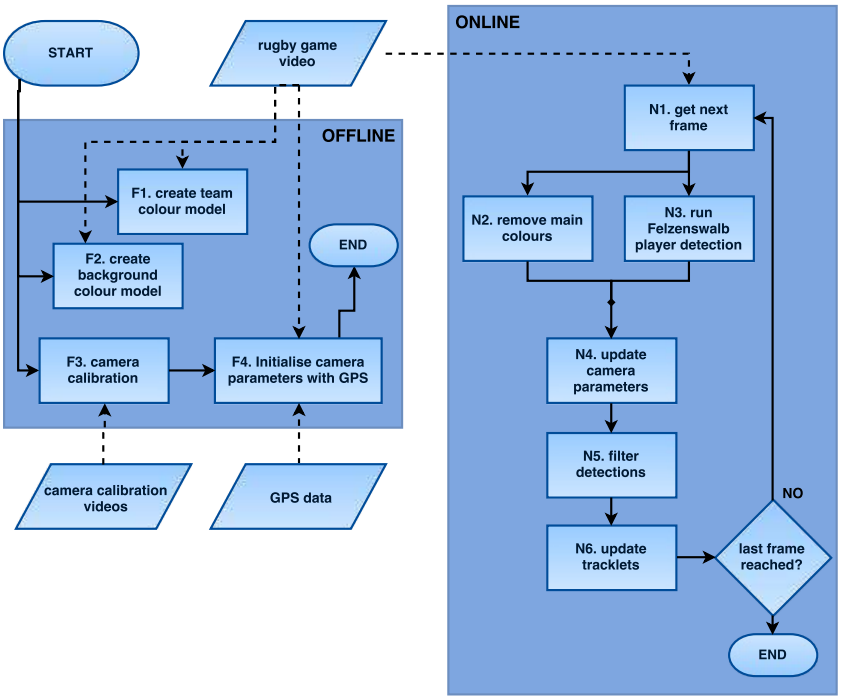

Figure 4.1: Flowchart of the proposed method. It shows the main processes, which are divided into two parts. An offline stage, in which colour models are created, and the camera parameters are determined, and an online stage in which the video is processed frame by frame, detecting and tracking players.

Section 4.3.3. Secondly, in order to find the ground plane positions of the players through time, the players themselves need to be found in each frame. To find these persons, the deformable parts model by Felzenszwalb et al. [11] is used, see Section 4.3.2. The last keypoint is the combination of detections done within each frame, such that a series of detections will refer to 1 person and 1 person only. For this tracking part we have decided to use Kalman filters to model the player dynamics, and to use a colour discrepancy to determine whether a player is similar to a previous detection. This results in the solution of the variables $\mathbf{r}_i$. Details are found in Section 5.4.

Globally, the method looks as follows (see Figure 4.1). There are two stages, an offline stage that preprocesses the videos, and an online stage that iteratively goes through the each frame of the video. The offline stage pre-processes camera data and starts off with the camera calibration. A background colour model is created to allow for separation of foreground and background. This is advantageous for both detection processing and the camera modelling. Based on an initial run of the player detector over the span of the video, detections are used to create a team colour model as well. Finally, the GPS information is used to initialise the camera parameters. Details are found in Section 4.2.

The online stage is the iterative process of going through each n^{th} frame and solving the state vector for that frame. For each frame, it starts with player detection using the deformable parts model of Felzenszwalb and the removal of the background colour regions. For the estimation of the camera parameters, a new method is proposed to estimate camera changes relative to a baseframe, instead of comparing each frame only to its predecessor.

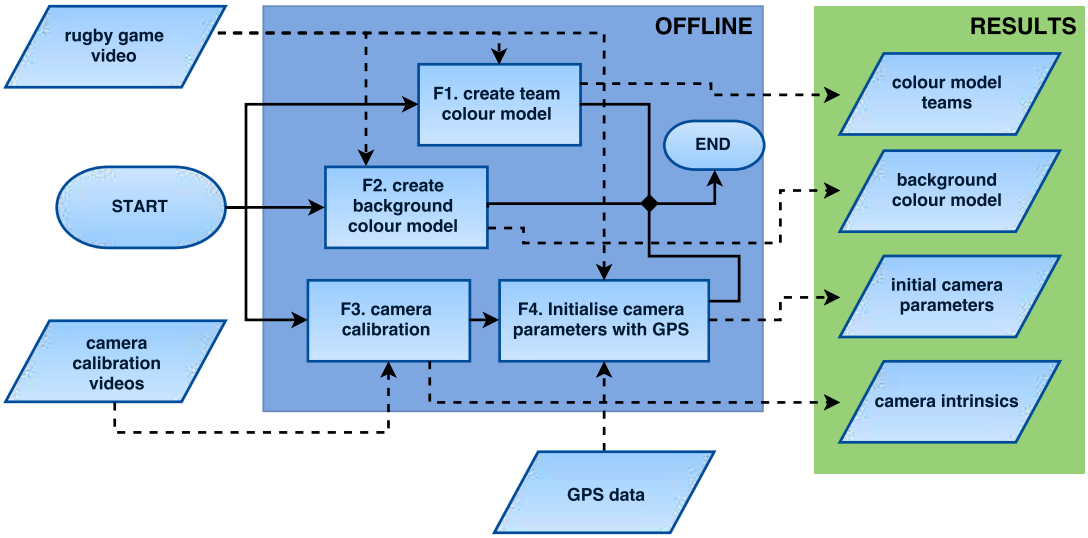

Figure 4.2: The extended flowchart for the offline stage shows the processes of the offline stage as well the outcomes and results.

The camera parameters are estimated by minimising the defined energy function, using a multi-start gradient descent algorithm. Details are found in Section 4.3.3.

The detections are filtered according to appearance, size and location, before the tracklets are updated. Detections which lie outside the field according to projection to the ground plane, or have an unreasonable height (in F^W), are disposed. The tracklets are updated by matching the remaining detections to the active tracklets. The cost of a detection-tracklet combination is based on the difference in appearance and the affinity of the detection with the dynamic model of the tracklet. The tracklets use the detections, projected onto a reference frame. Finally, each tracklet is reprojected onto the ground plane. Details of the online stage are found in Section 4.3.

4.2 Offline stage: pre-processing

In the offline stage we establish all the information necessary to start processing the video. As can be seen on the extended flowchart in Figure 4.2, there are four distinct process that can be identified: Creation of the team colour models, creation of the background colour model, camera calibration, and the initialisation of the camera parameters. The figure also shows the various outcomes of the processes shown in the green box on the right.

4.2.1 Team colour models

For each team we create a colour model based on histograms in the HSV colour model, using 15 and 8 bins for the H and S channel respectively. We exclude that V channel, since this channel is very sensitive to lighting changes. To add a distinctive black and white, we add two additional entries to the histogram. Black, with ranges $0-180$, $0-256$, $0-10$ and white with ranges $0-180$, $0-10$,$246-256$.

Experimentally, we have found that these settings give the most distinctive distributions when comparing two teams. This is largely due to the fact that most of the Dutch their opponents wear dark colours (black, dark blue etc.), while the referee wears a white shirt. Moreover, there are changes in illuminance throughout the video due to clouds, and colours seem to shift slightly when people were standing further away from the camera. Therefore, a colour space was chosen which was invariant to illuminance, but was also able to distinct (near) black and (near) white colours from others.

We run our player detector over a series of frames covering the whole range of the video, and select between 50 and 60 detections for each team. Of each of the detections a histogram is created, and we build two different colour models, which we compare in the results. The first model concatenates all the histograms, such that a new detection can be compared to all team player histograms.

The second team colour model combines all the individual histograms and determines for each team the mean and variance histograms. For both team we iteratively select the bins with the highest mean. The bin's distribution is compared between the two team using a two tailed t-test with unequal variance. For a t-test to work, the two distributions Gaussian, which is the case for the prominent colours, but unfortunately not for colours which are hardly present. A bin is selected when the null-hypothesis, stating that the distribution are similar, is rejected. The process is repeated until 15 bins are selected. Initial experiments showed the combination with the set HSV bin range 15 bins were sufficient to differentiate between the to teams. The final colour model is made by creating the mean vectors μ_1 and μ_2 of size (15×1) and the covariance matrix $\mathbf{S}$ of size (15×15), containing the covariances between the selected bins.

4.2.2 Background colour model

Before the online processing of the video, a background colour model is created. The model is able the separate the foreground from the background, where we define the surroundings as the background. This model is used to determine the areas where no detections should be present, and it indicates which areas can be used by the camera parameter update process.

A a set of frames is selected from the video over the entire range, and for each frame a colour histogram in the HSV colour space is made, using the same bin distribution as with the team colour models. There are two colour histogram created for each frame. The first histogram records the ground plane colours. For each of the videos, an average horizon level is determined above which none of the colours will be registered. Additionally, the mask prohibits the lower 10% of the image to be used for the background, because of the crowd being visible in the lower image region during significant parts of the video. The second histogram covers the sky and the scenery behind the field, which is typically the upper 20% of the image.

The final background colour model is found taking the 10 bins with the highest mean from the combined normalize lower region histograms, and 7 bins from the combined normalised upper regions histograms.

4.2.3 Camera calibration

The camera is placed on top of a tripod, and occasionally zooms in and out on parts of the game such as scrums, conversions and fouls, and the camera is modelled as a pinhole model. The parameters needed in order to transform a 3D world point to the image plane are given by the camera matrix $\mathbf{P}$. This transformation matrix is subject to change when the camera either rotates or zooms. Therefore, the camera matrix is parametrized on these parameters, adopting the notation from [24]:

$$\mathbf{P}^{p,t,z} = \mathbf{K}^z[\mathbf{R}^{p,t} \mid -\mathbf{R}^{p,t}\mathbf{C}] \tag{4.2}$$

where $\mathbf{K}^z$ refers to the camera intrinsics,

$$\mathbf{K}^z = \begin{pmatrix} f_x^z & 0 & p_x^z \\ 0 & f_y^z & p_y^z \\ 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

$[\mathbf{R}^{p,t} \mid -\mathbf{R}^{p,t}\mathbf{C}]$ are the extrinsics of the camera. Note that the superscript t refers to tilt, as in the rotation around the camera's x-axis, and not time. f refers to the focal length in either x or y direction of the camera's lens and (p_x, p_y) is the principal point of the image. We model the distortion up to 5 degrees, and describe it by the parametrized vector $\mathbf{D}^z = [\kappa_1^z, \kappa_2^z, p_1^z, p_2^z, \kappa_3^z]$. Whilst the rotation matrix changes when either panning or tilting, it does not change when zooming. The opposite is true for the intrinsic matrix and the distortion, which are independent of the scene viewed, but changes with varying zoom level. The translation vector C, defined as the location of the camera, expressed in the world coordinate system F_W is static for our problem because of the tripod.

To perform the calibration sequence for a single focal setting, we use the method proposed by Zhang [38]. This method allows us to find the intrinsics $\mathbf{K}$ at a given zoom level z. Besides the intrinsics, this method will also result in finding the distortion coefficients for radial and tangential factors. This eliminates the 'fish-eye', 'barrel', and 'moustache' effect, and corrects the image for the image plane not being parallel to the camera's lens.

The calibration data for the entire zoom range is found by repeating the single focal setting calibration at multiple zoom levels, starting at z_0, up to the maximum zoom level z_{max} with linear intervals. For all zoom levels in between, we find the values by linearly interpolating each of the variables in the intrinsics and distortion vector. This approach is similar to the approach by Taketomi [28]. The results are found in Section 5.1.

4.2.4 Camera initialisation

At the start of the video, a series of subsequent, still frames are selected for the initialisation of the projection matrix $\mathbf{P}^z$. We assure that during these initial frames there is no zooming or rotating. To determine the extrinsics, the GPS data is introduced. The GPS data is time-synced with the video, and the GPS data is manually matched with the image location of the feet of the respective player for a number of frames.

The pairs of GPS and computer vision detections are used to find the projection matrix $\mathbf{P}^{z_0}$. Since there is an unknown z_0, we solve for both the intrinsics and the extrinsics.

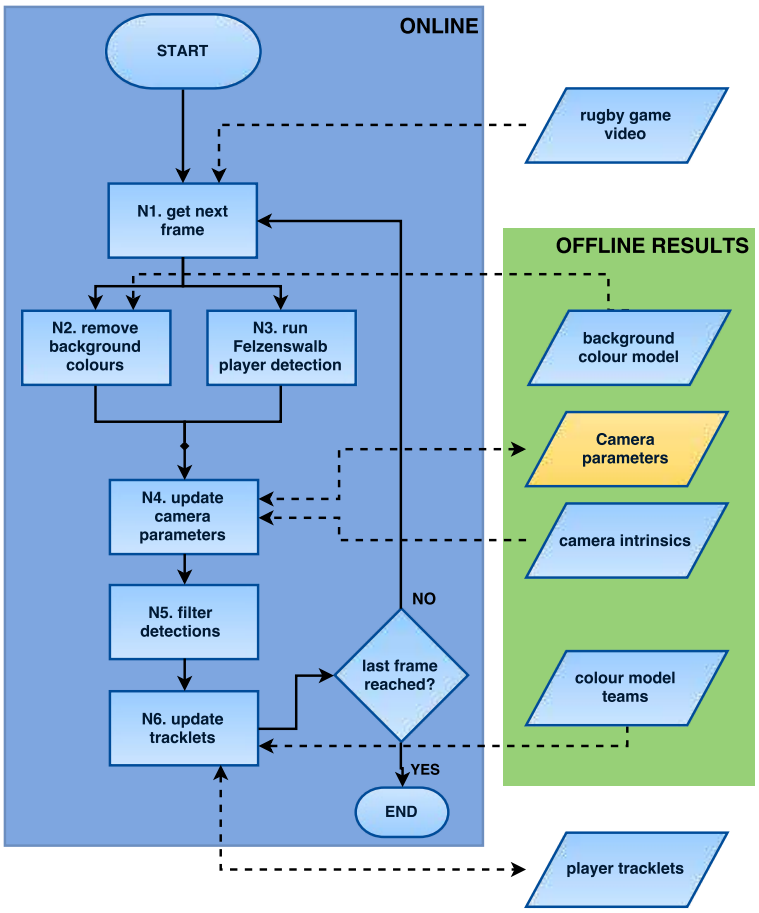

Figure 4.3: The extended workflow diagram of the online stage. Here, the results of the offline stage are shown as well, as well as the interaction with them.

Iteratively, we solve for all the extrinsic parameters of $\mathbf{P}$ by varying the zoom value z within the range reasonable range. At each iteration, the PnP problem is solved using the iterative Levenberg-Marquardt optimisation algorithm, and the reprojection error is recorded. The zoom level z is varied according to this reprojection error according to the gradient descent method, where the step size is decreased only if the direction has changed. To prevent ending up at a local minimum, the search is repeated for at 3 different begin positions $z_{start} \in \{1, 5, 8\}$.

4.3 Online stage

The second stage of the methodology is the online stage in which the video is processed at every 2 frames. The videos we use are recorded at 25 fps, resulting in a process amount of 12.5fps. This will yield us enough information to determine the whereabouts of the players, and it gives in increase in process speed of a factor 2.

An extended overview of the processes of the stage are given in Figure 4.3, along with the results of the offline stage, and the outcome of the stage itself. The stage starts with selecting

a new frame, after which the removal of background colours and the person detection are performed in parallel. Using the found background area, as well as player locations in the image, the camera parameters are updated. Then, before updating the tracklets, the detections are filtered and their respective team is determined. Finally, the resulting detections and camera information is fed to the update tracklet process, in which tracklets are expanded with the new detections, initiated when no matching tracklet was found or terminated accordingly. Throughout this section, we will describe each part of the process individually.

4.3.1 Background colour subtraction

To prepare the camera update step, the areas with colours that coincide with the background colour model are saved in the background mask. Additionally, a number of morphological processes (dilation and erosion) are applied to the masked image to create consistent areas of background and foreground.

4.3.2 Felzenszwalb player detection

For the person detection, the decision has been made to use the framework of Felzenswalb [11], an object detector with a deformable parts model. The framework is based on the histograms of oriented gradients (HOG) as described in [6]. Although it is not known for its speed, it generally outperforms other methods in terms of precision and recall [9]. One of the big advantages of Felzenszwalb's framework is that it its model is based on the deformability of objects. In case of people this is especially convenient, since they show large variation in terms of articulation. Other advantages are its broad use and numerous implementations available. Moreover, there are pre-trained models available for several object classes, such as pedestrians, of which we will make use. Although this model is not based on large articulation made by our sportswomen, it will serve as a good first benchmark. Although the articulation of the limbs of rugby women is larger is many of the cases, e.g. sprints, tackles, scrums, many of the frames contain poses in which they have an upright, pedestrian like pose.

Each of the detections found by the detector is given a confidence level. During detection, an image pyramid and associated feature pyramid is created at which, with a sliding-window approach, the response to the root filter at every location is measured, see Figure 3.5. At twice the resolution of the root filter, the part responses are calculated sequentially and only at points for which the cumulative response of the previous parts and the root is above a threshold, see Figure 3.6.

The total score of a location is formulated as:

$$\text{score}(x_0, y_0, l) = R_{0,l_0}(x_0, y_0) + \sum_{i=1}^{n} D_{i,l_{0-\lambda}}(2(x_0, y_0) + v_i) + b \tag{4.4}$$

The total score for a single component m , at location (x_0, y_0), found at pyramid level l, is the response of root filter at that location, plus the responses of the transformed parts. These parts are allowed to deviate from their anchor position defined by the part model

(v_i). Lastly, the score is increased with the component bias b. The part scores D are given by:

$$D_{i,l} = \max_{dx,dy}(R_{i,l}(x + dx, y + dy) - d_i \cdot \phi_d(dx, dy)) \tag{4.5}$$

where the displacement cost is defined by $d_i \cdot \phi_d(dx, dy)$, with d_i being the model defined quadratic displacement function and $\phi_d(dx, dy) = (dx, dy, dx^2, dy^2)$. The location scores are calculated for each component individually, and the component with the highest score is chosen as the final score for that location. After the frame is processed, the detections are enhanced by performing bounding box predictions and performing non-maximum suppression. The final score is returned as the confidence level of the detection. The model used for our process uses the *person_final* model which was trained with data from the VOC2010 database.

4.3.3 Update camera parameters

Most recent work that focusses on multi object tracking with moving or rotation camera's in the image ground plane can be distinguished into two methods. Pose estimation is either based on finding a marker, and using the known 3D points to solve the PnP-problem, or by tracking ground plane feature points, of which their 3D correspondence is set by the found (initial) camera pose.

In the second case - the markerless camera pose estimation, the camera parameters are generally estimated sequentially, where the results of the previous frame are used as input for the next frame. For each estimation however, there is a small error in the homography between the frames, the location of the feature points, as well as in the reprojection error. These errors are carried on through the subsequent frames, and the accumulation of transformations is known to result in an accumulation of errors as well, and hence a possibility of drift. This is generally avoided by adding a filter, such a Kalman filter or a moving average filter, which is able to mediate the error propagation to a large extent [5].

Sports videos that are recorded from a single point of view, and follow the ball around during the game, have the advantage that the camera parameters are recurring. Over time, the camera typically rotates and zooms from a center view towards a view near one of the goals or try lines, after which it returns back to its initial state (defined by its position, orientation and zoomlevel) or a state close to it. This recurrence of the camera parameters can be used to our advantage.

In order to prevent errors from accumulating throughout the video, we propose to use a set of reference frames, called *baseframes*. A baseframe is defined as a frame with a low error with respect to reprojection to which other frames relate their rotation and zooming. By means of feature matching, the changes in rotation and the new zoom value can be found, as long as there are enough keypoints (features) that can be matched. Over time, these baseframes form a set of frames spread over the traversed camera state ranges, such that the image overlap between any of the different baseframes is smaller than a given threshold.

To derive the camera parameters at any part of the video, we propose an algorithm, which is comparable to the method recently proposed by Taketomi et al. [28]. They propose a continuous pose estimation solution for a free moving, rotating and zooming camera in which at every frame, a KLT tracker is used to track natural features, and a fiducial, square marker

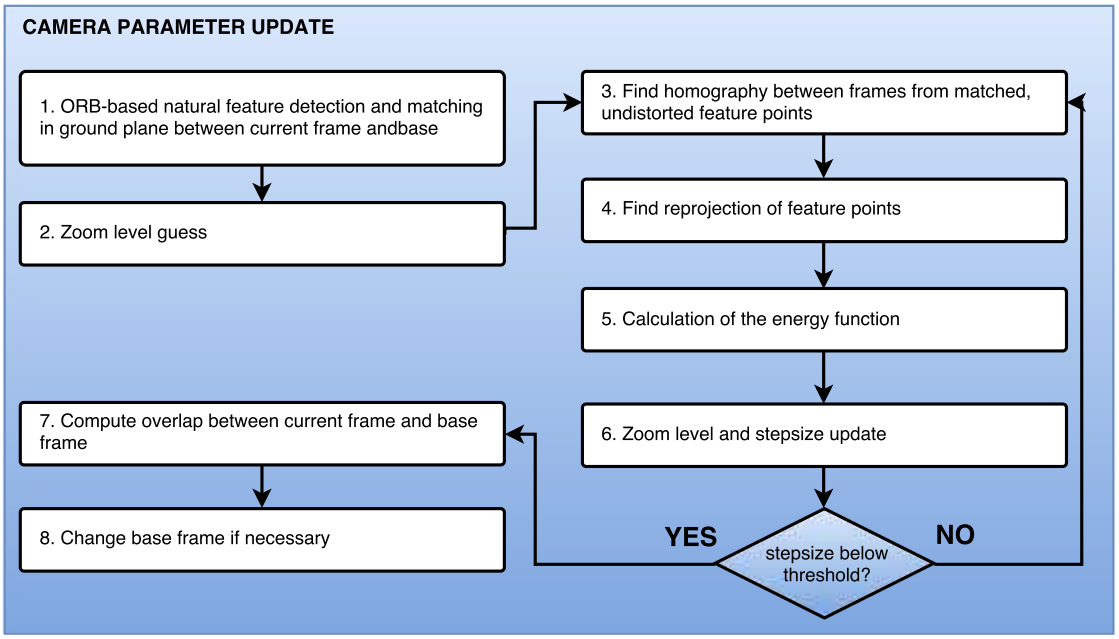

Figure 4.4: The steps involved during the update camera parameter process.

is detected. The KLT features are only used in between successive frames, to prevent large errors, and to promote continuous, smooth movement of the camera. A new energy function is defined, that records (1) fiducial marker reprojection errors, (2) reprojection errors of the tracked features based on the epipolar constraint and (3) the constraint of zoom continuity. At each frame, a zoom value is estimated, and energy function is calculated. Based on a Levenberg-Marqaurdt optimisation, the zoom value is iteratively changed, and the energy function is minimised.

We adopt this method to fit it to our challenge. The largest difference is the non-sequential relation between frames. At each frame, we determine the camera parameters using the active baseframe as the reference frame for tracking natural features, and there is no fiducial marker present. Initially, attempts were made to perform detection of the various lines and corners on the field, but due to the bad quality of the field and the viewpoint of the camera, these were hardly identifiable, both by eye and detectors. Other changes include the use of an ORB feature detector and descriptor, as well using the homography error instead of the error from the epipolar geometry. An overview of our adopted algorithm is given in Figure 4.4.

During the playback of the video, each subsequent frame is used to determine the camera rotation, relative to the active baseframe. The frame used during initialisation to determine the starting camera parameters, is set as the first baseframe. Upon initialisation of a baseframe, keypoints are detected on the undistorted image using the background image mask. For the resulting ground plane keypoints their 3D correspondences are calculated. Their back projection from the 2D image to the 3D world is possible, because of the ground plane constraint of the keypoint, i.e. $Z^W = 0$.

The algorithm starts by detection and matching natural features in the frame. Instead of using a KLT tracker, which is light sensitive, features are tracked using the ORB feature

detector and descriptor [23]. ORB is an efficient alternative to both the SIFT [19] and SURF [1] keypoint detectors, and has a reduced complexity and an improved performance. The found keypoints are matched to the keypoints of the active baseframe. It is important that only static background features are detected, and that the detected points lie on a single plane. Therefore, the irregularities on the field, the lines and the bottom of the goal posts are perfect candidates keypoint regions. The background colour model selects these regions, as well as other regions around the field which are picked up by the sky region part of the model. To prevent keypoints being taken from any part but the field, keypoints can only be found from the bottom 50% of the image.

Iteratively, a solution is found for the pose estimation, by minimizing the error function. The error function used is defined as:

$$E = \omega_{rp} eRepr + \omega_H eH + \omega_z eZoom + \omega_T eT \tag{4.6}$$

The total error E is the weighted sum of the reprojection error of the keypoints from their 2D-3D correspondences, and the reprojection error of homography between the keypoints. Additionally, a constraint on the zoom continuity and the camera location is added.

An iteration starts by undistorting the matched keypoints with the intrinsics from the zoom level guess. The undistorted keypoints on the baseframe and the target frame used to find the homography $\mathbf{H}$ between the images,

$$s_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{4.7}$$

such that the error defined as

$$\sum_i \left[\left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \right] \tag{4.8}$$

is minimized. It is important to acknowledge here that $\mathbf{H}$ is scaled, such that h_{33} is equal to 1. To prevent outliers or wrongly matched keypoints to have a negative effect on the optimization, a RANSAC procedure is added. The homography error eH only takes into account the points considered inliers by the algorithm.

The inlier keypoints from the target frame, together with the 3D correspondences given by the baseframe are fed into the iterative PnP-solver as explained in Section 3.1, resulting in the rotation matrix and location vector. The minimised reprojection error $eRepr$ is found by projecting the 3D points onto the image plane, and summing the total error.

The zoom continuity constraint is defined:

$$eZoom = \frac{1}{f_x^z}(z_{guess} - z_{prev})^2 \tag{4.9}$$

equal to the one by Taketomi et al. [28], and the translation continuity constraint is defined as the Euclidean distance between the found and the initialised camera location.

The final error function is compared to the lowest available error, and replaced if lower. For the next iteration, the zoom level is determined based on the current zoom level and the zoom level corresponding to the best result. We use a simplified gradient descent algorithm, where the step size is kept equal unless the search direction changes, at which it is multiplied by 0.7. The search direction is changed when the best solution was found at a zoom level located at the opposite direction of the current solution. The method is stopped whenever the step size becomes smaller than 0.05. To prevent ending up at a local minimum, the process executed at three different initial zoom values, by adding an offset to the initial guess. These offsets are set to $\{-1, 0, 1\}$.

In the last two steps, it is checked whether the baseframe has enough overlap with the target frame. The overlap between two frames is determined by the homography. The corners of the target frame are projected onto to baseframes, and the amount of overlap is calculated by finding the intersection of the rectangle and quadrangle. The final overlap is given in two variables; the amount of overlap with respect to the baseframe, i.e. the intersecting area divided by the baseframe's rectangle, and the overlap with respect to the target frame, i.e. the intersecting area divided by the area covered by the target's frame reprojected rectangle.

Since the feature matcher works best when points are distributed over a larger part of the image, we put a threshold of 70% on the amount of overlap that has to be present. Acknowledging that our homography will contain some noise, we reconsider our baseframe when the overlap is below the threshold 3 times in a row. If this happens, the frame with the error from the last 3 frames is considered as a new baseframe candidate. This candidate is compared to the other baseframes that have been used in the past. If there is an older baseframe with enough overlap, and with a cumulative error then the candidate, the old baseframe will be set to active, and subsequent frames will be compared to this frame. This cumulative error E_{nb} is the resultant error between the frame and its baseframe E_t plus the error between the used baseframe and its own reference frame E_b i.e.

$$E_{nb} = E_f + E_b \tag{4.10}$$

4.3.4 Team selection of detections

The detections are generic and only acknowledge the presence of a pedestrian-like object within the given boundary box. Not all results of the object detector are of interest however. A large number of false positives are present - the condition when erroneously a player has been detected, but in reality there is no player. To suppress the number of false positives the found detections are filtered before feeding them to the tracking algorithm. Due to the articulated nature of the poses of the rugby players, a relatively low confidence level threshold is set for the object detector. This allows for more extreme poses to be recognised and detected. The big drawback of this approach however, is the increase of false positives A series of detection filters is added to address this problem. These are:

1. filter by background colour model
2. filter by projected height
3. filter by projected location
4. filter by team colours

The background colour model filters the detection by iterating over each detection, and counting the amount of earlier assigned background pixels within it. When the ratio of background pixels to total pixels is larger than 50%, it is believed the detection was a false positive, and the detection is discarded.

Knowing the camera parameters, it is possible to make a prediction of the 3D location the players. Using the assumption that the bottom of a detection corresponds to the feet of the person the mapping from the image plane to the world (in 3D) can be made (see bottom paragraph, Section 4.3.5). Assuming that the (X_W, Y_W) ground plane coordinate for the feet is equal to the (X_W, Y_W) coordinate for the top of the detections, and that the top of the bounding box corresponds to the top of the person, allows us to derive the height of the detection in the world. The field size of a rugby field is 100 by 70 metres from try line to try line, with an area behind the goal line of at most 20 metres. Therefore, we put a cap on the projected location of the detections at 65 metres in X_W direction (both positive and negative), and 35 metres in the Y_W direction, also both positive and negative). For the detection height, a boundary is set such that $1.0 <$ detect height < 2.5. The height bound is not a tight one, but this allows for detections of a kneeling person, as well as some bounding box leniency towards the detector.

The final filter, specifies the team to which the detection belongs. The two different colour models are processed as follows. In case of the comparison with all histograms, a histogram is made in the same colour space and with the same hist bin configuration as the model. Then, for each histogram in the model, the histogram distance is calculated. The total score is defined by averaging over the n best scores. By using only a subset of the available histogram, it is expected that the colour model becomes invariant to viewpoint. Depending on the viewpoint, the colour distribution can be different, such as large numbers or colour patches on the back of the shirts. The distance between two histograms is given by the OpenCV implementation of the Chi-Squared distance [2], denoted as

$$d(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i)} \tag{4.11}$$

A detection is assigned to one of the teams when the resulting error is below the set threshold. In the results 5.3, an analysis on the threshold and the quality is given.

For the selective colour model, the relevant bins as defined by the team's model, are selected from the detection's colour histogram, and put into the measurement vector μ. The m dimensional vector is compared to either of the team's colour model using the Mananalobis distance. This is given by:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \mu)^T \mathbf{S}^{-1} (\mathbf{x} - \mu)} \tag{4.12}$$

This distance is based on dissimilarity between the measured vector, and the established model for the player. The dissimilarity can be seen as a weighted Euclidean distance from the mean vector to the measured vector, where the weights are determined by the covariance matrix, such that the resulting distance gives the number of standard deviations the data point deviates from the mean point defined by the model. A detection is assigned to a team whenever the resulting distance is below the set threshold, which is put between 3 and 8.

Back projection image point to 3D world

For each detection is it assumed that the lower edge of the bounding box corresponds with the ground, meaning the ground plane z coordinate (in F^W) is equal to 0. This assumption is necessary, because the back projection maps a 2D coordinate to a 3D one, i.e. an additional dimension is added. Following the notation in Figure 3.4, every point on the line through the camera centre and the object's world coordinate $\mathbf{X}^W$, is projected onto a single image coordinate. The distance of the world coordinate to the camera centre is given in camera model by the scaling parameter λ. Rewriting the model, assuming there is already corrected for distortion, gives

$$\begin{aligned}
\lambda \begin{pmatrix} x^{im} \\ y^{im} \\ 1 \end{pmatrix} &= \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix} \\
&= \mathbf{K} \begin{bmatrix} \mathbf{R} & -\mathbf{RC} \\ \mathbf{0}_3^T & 1 \end{bmatrix} \begin{pmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{pmatrix} \\
&= \mathbf{KR} \begin{pmatrix} X^w \\ Y_w \\ Z^w \end{pmatrix} - \mathbf{KRC}
\end{aligned} \tag{4.13}$$

This finally gives us

$$\begin{pmatrix} X^w \\ Y^w \\ Z^w \end{pmatrix} = \mathbf{C} + \lambda \mathbf{R}^{-1}\mathbf{K}^{-1} \begin{pmatrix} x^{im} \\ y^{im} \\ 1 \end{pmatrix} \tag{4.14}$$

Equation 4.14 shows that the scaling parameter is necessary to find the backprojection. Using the assumption that people stand on the ground, gives us $Z^W = 0$, and allows us to solve for lambda:

$$\lambda = \frac{Z_w - c_3}{z_3} \tag{4.15}$$

where $(z_1, z_2, z_3)^T = \mathbf{R}^{-1}\mathbf{K}^{-1}\mathbf{x}^{im}$, and c_3 the camera position.

4.3.5 Tracklet update

Player occlusion occurs due to the viewpoint of the camera, missing detections from the player detector, and wrong team assignments are reasons tracklet assignment won't work when combining detections into tracklets solely from detections. We assign a Kalman tracker to each tracklet, which estimates the position and velocity of the tracklet, and keeps track of the size of the detections.

The time history of of the Kalman filter represents a series of estimated and corrected locations of a player. For each Kalman filter, assigned to a single tracklet, the following dynamics are used:

$$\begin{aligned} s_t^x &= s_{t-1}^x + v_{t-1}^x \Delta t \\ s_t^y &= s_{t-1}^y + v_{t-1}^y \Delta t \\ v_t^x &= v_{t-1}^x \\ v_t^y &= v_{t-1}^y \\ \delta^x &= \delta_{t-1}^x \\ \delta^y &= \delta_{t-1}^y \end{aligned} \tag{4.16}$$

The location of the player, given by s_x and s_y for the x and y-direction respectively, is updated by adding its velocity v_x and v_y to its current position, split in the x-and y-direction. We model the velocity of the player to be constant, but it is allowed some deviation through the noise parameter. Since the changes in location are bound, the velocity changes are bound as well. Additionally, we track the size of the detections in the image plane in both directions, denoted by δ^x and δ^y respectively. We refer to the Kalman filter model as shown in Equation 3.14. Plugging in our dynamic model into the Kalman filter gives the transition matrix $\mathbf{F}$ and measurement matrix $\mathbf{H}$

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

$$\mathbf{H}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.18}$$

We determine the control error covariance matrix $\mathbf{Q}_t$ by going through our GPS data for variations in location and instantaneous velocity. The control error for the detection size is derived from testing. By manually assigning detections to tracklets, information on the variance of the bounding box sizes is found. The resulting covariance matrix is modelled as a static matrix and is kept the same for each tracklet.

The measurement covariance matrix $\mathbf{R}_t$, not to be confused with the camera rotation matrix, is defined as a non-static variable, which is dependent on the distance of projected location of the detection with respect to the camera. When the bounding box of a detection has a small offset in pixels in the vertical (image) direction due to noise or errors from the object detector, this results in an erroneous back projection. The camera has a relatively low angle with the line through the horizon. As a consequence, the field width is - in case of minimum zoom - covered by at most 400 pixels. The width of the field is not linearly divided over the pixels that cover it. The distance in pixels from the midpoint to the far sideline is significantly less then the distance to the near sideline. As a consequence, bounding boxes with their lower border closer to the bottom of the image have smaller back

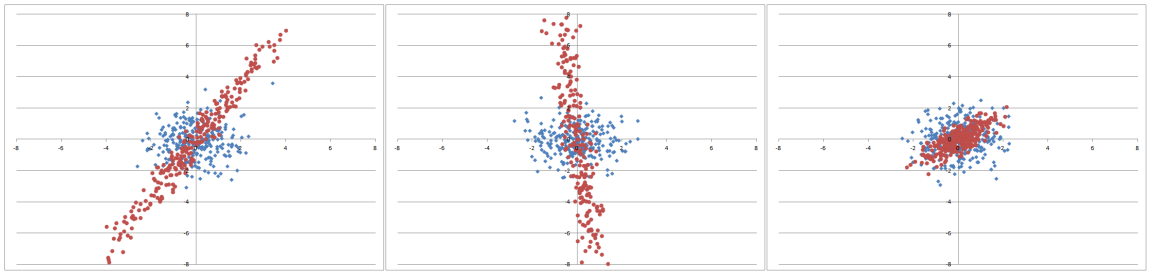

Figure 4.5: Covariance examples of the measurement covariance matrix for simulated data points. The blue dots correspond to the distribution of the unit covariance, whereas the red dots refer to the location and rotation adapted covariance. The scale matrix used was $diag(0.48, 0.1y_t^*)$.

projection errors than detections having their lower border further away from the bottom of the image.

To adjust for this non-static behaviour of the error in the look along-direction, a location dependent covariance matrix is defined. We assume that the x and y coordinate of a detection in the image plane are not correlated, but that its projection onto the ground plane is. This correlation is in the same direction as the line from the camera centre through the projected point. The resulting covariance found for the location is a scaled and rotated version of a diagonal matrix, such that the top part of the covariance matrix $\mathbf{R}_t$ is given by

$$\mathbf{R}_t = \mathbf{R}^p \mathbf{S} (\mathbf{R}^p)^T \tag{4.19}$$

Here, $\mathbf{R}^p$ refers to the rotation matrix derived from the pan of the camera. Because our videos start with a view which in which there is no pan,we use the pan value from the found camera parameters. The diagonal scale matrix S gives the variance for the case of no pan. It is defined as

$$\mathbf{S} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & y_t^* \sigma_y^2 \end{bmatrix} \tag{4.20}$$

where y_t^* is the distance of the projected point to the camera, and σ_x and σ_y refer to the standard deviation the uncorrelated x and y direction. Examples of the covariance spreads are shown in Figure 4.5.

Tracklet affinity

For each of the detections that are matched to a team, we determine their affinity with the tracklets by means of their dynamics, i.e. motion agreement, their colour resemblance, and the image plane size of the detection. We set up a cost function c as the weighted sum of the affinities.

$$c = \omega_d c_d + \omega_c c_c + \omega_c c_s \tag{4.21}$$

Where ω refers to the weight and c_i to the affinity in terms of i, given for dynamics(d), colour (c) and size (s). We only match detections assigned to the Dutch team with Dutch tracklets, and opponent detection with opponent tracklets. During tracking, each tracklet predicts the next location of the player it's following according to the Kalman model, and compares the resulting distribution with the distribution we get from the back projection of the detection. The tracklet's dynamics are given by the state vector, of which we exclude the velocity and size components. The *a priori* error covariance matrix $\mathbf{P}_{t|t-1}$ states the current error of the prediction which is used as well. The detections dynamics are defined by its location and size, as well as the measurement covariance matrix $\mathbf{R}_t$ of the tracklet. This way, we can include the reprojection error in the affinity calculation.

The dynamics of both tracklet and detection are captured in two multivariate normal distributions(denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$). The amount of overlap between them is found by the Bhattacharyya coefficient, which is derived from the Bhattacharyya distance:

$$D_B = \frac{1}{8}(\boldsymbol{\mu}_d - \boldsymbol{\mu}_{tr})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_d - \boldsymbol{\mu}_{tr}) + \frac{1}{2} \ln \left(\frac{\det \boldsymbol{\Sigma}}{\sqrt{\det \boldsymbol{\Sigma}_d \det \boldsymbol{\Sigma}_t}} \right) \tag{4.22}$$

which is a similarity measure of the two distributions, such that $D_B = -ln(BC)$, the negative natural log of the overlap ratio. In the equation $\boldsymbol{\Sigma} = \frac{\boldsymbol{\Sigma}_d + \boldsymbol{\Sigma}_{tr}}{2}$ is the combinatorial covariance. This distance represents the similarity between the two distributions, where the first part is the Mahanalobis distance. This calculates the distance from the detection mean to the tracklet mean, expressed in amount of standard deviations by the mean covariance of the two. The second part compares the variances of both distributions. It is very likely that the distributions are very different from one another, since the variance direction of the tracklet is most apparent in the direction of travel, whereas the variance direction of the detection is most apparent in the direction connecting the back projection and the camera location. When we leave out the second part of the Bhattacharcy distance, this will not penalise the different shapes of distributions, but will incorporate the variance of not only the tracklet, but also the detection. Not using the variance similarity reduces the cost to the Mahanalobis distance only. To emphasise bad scores, we put the resulting affinity in an exponential function. Therefore, the final dynamic cost function, leaving out the $\frac{1}{8}$ scalar multiplication, becomes:

$$c_d = e^{M_d/2} - 1 = \sqrt{\Big(\boldsymbol{\mu}_d - \boldsymbol{\mu}_{tr}\Big)^T \boldsymbol{\Sigma}^{-1} \Big(\boldsymbol{\mu}_d - \boldsymbol{\mu}_{tr}\Big)} \tag{4.23}$$

For the comparison of the colour distributions, we compare the latest matched detection of the tracklet, and compare its colour histogram to that of the detection. We compare the detections using the same HSV histogram as we did for the two colour models. The histograms are compared to one another using the Chi-Square distance, given by Equation 4.11. This is given as the colour cost c_c. Again we put the result in an exponential function, to gain more control on the affinity output, as well as to prevent larger errors from a single cost component to be compensated by lower scores from the others.

The size cost function takes a similar form as the dynamic cost function. This time, we do not incorporate a covariance on the detection size, and only use the tracklet error covariance matrix. We use the Mahanalobis distance to get the distance between the detection and

the modelled size. The size of the detection is however very dependent on the zoom level as well as the distance of the player to the camera, and initial results of the detector show that for a single person, the detections throughout the various frame have the tendency to vary significantly, up to 8 pixels in some extreme cases. Therefore, we have decided to put a relatively lenient constraint on the cost of size inequalities between tracklet and detection. The final cost function for the size affinity is $c_s = e^{M_s/2} - 1$, with M the Mahanalobis distance as expressed in Equation 4.12.

We allow at most 1 detection to be assigned to a tracklet, and we allow at most 1 tracklet to be assigned to a detection. To determine which tracklet belongs to which detection, two Hungarian matching algorithms are used, on for each team, with the matching cost equal to our cost function results. The resulting matches are used to update the tracklet with the matched detection, only if the matching cost is below a set distance threshold.

Tracklet initialisation, growth and termination

The tracklets are updated with the information from their matched detection. Using the information of the camera pan, the measurement covariance matrix is set, and the Kalman filter is updated.

When there are detections which are not matched to a tracklet, either because there are too many detections, or because the threshold was not met, the detection will be used to initialise a new tracklet. Upon tracklet initialisation, the Kalman filter will be set up with the detection's location and covariance. The velocity is initially set to 0 with a large uncertainty, set in the error covariance matrix. The tracklet is set to a state called *working* and *unreliable*, referring that the tracklet is in use and initialised, but is not reliable because there's too little information available. After 5 iterations, the minimum of 3 found detections has to be met, otherwise, it will be terminated. It enough detections have been matched, it will change its state to *reliable*.

In case there is no detection available for an existing tracklet, the tracklet will predict the new location and leaves this uncorrected. This is allowed to happen at most 10 frames consecutively, otherwise the tracklet is terminated and its state changed to *non-working*. Because we deal with videos in which not all players will be visible at all times, we do stop our tracklet when its location is outside the camera's view for more 6 consecutive frames (corresponding to roughly 0.5s). The same holds when a tracklet moves outside of the field.

Chapter 5

Experimental results

The experimental results presented in this chapter come from the video of the first match of the tournament, named *Game 1 vs Maple leafs.MP4*. Additionally, we make use of the provided GPS data by Johan-Sports as well as the calibration videos. To demonstrate the capabilities and performance of our methodology, a number of experiments are performed and explained in this chapter. We start off with the results of the camera calibration, after which we validate the camera model results. Finally, the tracklets are introduced and we analyse the results by comparing them to the GPS tracks.

5.1 Camera calibration results

The camera used for filming the rugby matches was acquired for a day, such that a series of calibration videos could be taken. A calibration video consists of a short sequence of about 30 seconds, in which a moving and rotating chessboard pattern is filmed. The chessboard is divided in 9×7 squares, each of size 115×115 mm. The entire, continuous, zoom range of the camera was divided into 32 zoom levels, coinciding with the amount of zoom level indicator positions on the camera display. At every 4th zoom level, a calibration video was made and during a single calibration video, there was no zoom.

For each of the calibration videos, a camera calibration was performed as described in Section 4.2.3, resulting in the intrinsics at the given zoom level. The individual results were put together, and for zoom levels in between calibration, the results were linearly interpolated with the 2 surrounding data points. These results are shown in Figure 5.1.

Four graphs depict the 9 parameters that define the intrinsics of the camera for the full camera range. In the top row, distortion coefficients for radial distortion ($\kappa_1,\ \kappa_2,\ \kappa_3$) and

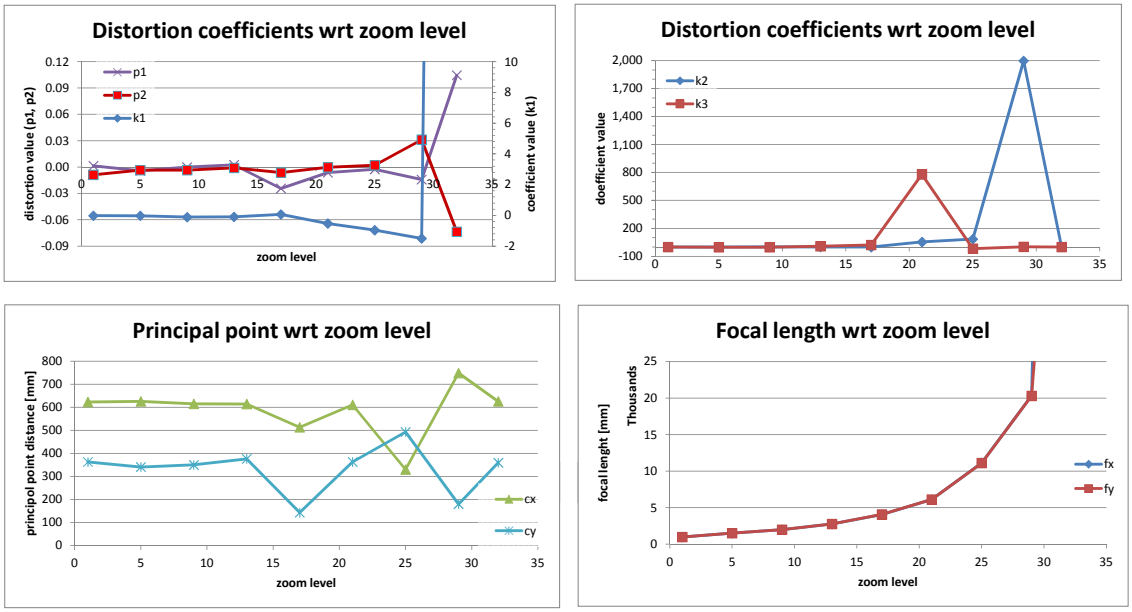

Figure 5.1: The initial derived intrinsics for the camera. It shows the equality between the horizontal and vertical focal lengths ($f_x = f_y$).

tangential distortion are shown (p_1, p_2). Up to approximately zoom level 17 the distortion coefficients are all relatively constant. After zoom level 17, the pattern is disrupted, and all of the parameters start to behave erratically. A similar observation can be made about the principal point, depicted in the bottom left in Figure 5.1. Varying around (623,363) until zoom level 15, the result of zoom level 17 gives a disruptive result, after which at zoom level 21, the system is restored, only to be disrupted again at for the remainder of the zoom range. The focal length however, behaves in a continuous exponential manner, where the focal length in x- and y-direction completely overlap. This means that the pixels are square, and in our model $f_x = f_y$, reducing the degrees of freedoms of the system by 1, and thus reducing complexity.

An explanation for the erratic behaviour of the camera parameters from zoom level 17 onwards is partially found in the results of the calibration at that zoom level. The calibration video used was not as elaborate as for the other zoom levels. It did not contain any frames in which the chessboard was held and rotated in the bottom left corner of the image plane. Because the distortion effects are most apparent near the edges of the image plane, missing frames depicting the chessboard near any one of the corners could have caused the calibration to bias towards different results.

A second explanation is that the reprojection error object points back onto the image plane. As a visual validation method, a back projection is made of the chessboard axes as well as a cube spanning a large part of the chessboard's plane. For back projections for zoom levels of 17 and upwards, the projection starts showing improper deformations. Some examples are shown in Figure 5.2.

To use the data during the online phase of the algorithm, three adjustments are made. Firstly, the measurement at zoom level 17 is removed, and new values are estimated based on the results of the surrounding zoom levels. A quick inspection shows that there is no

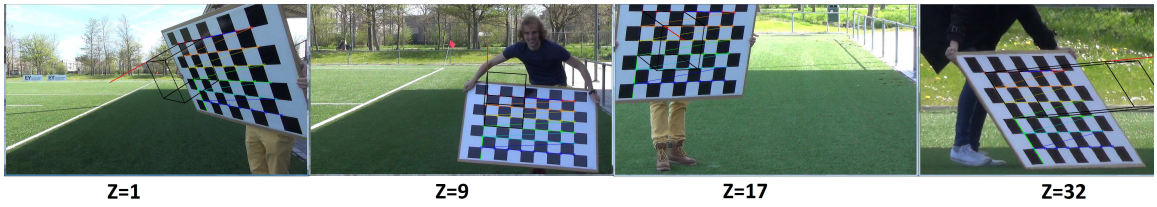

Figure 5.2: Back projection results from the camera calibration, showing a cube drawn onto the chessboard plane. It shows incorrect projections in the last frame.

significant increase in the average back projection error, changing from 0.639 to 0.705 pixels on average. The second adjustment is in the focal length. We set the focal length in x-direction equal to the focal length in the y-direction, and we fit it to an exponential function, by iterating over the two unknowns until while maximizing the coefficient of correlation. The resulting equation $923e^{\frac{z}{11.6}}$ has a residual of 0.9981, justifying the fit. Lastly, a cap is put on the maximum reliable zoom level, which is put at 20. These final changes are shown in Figure 5.3.

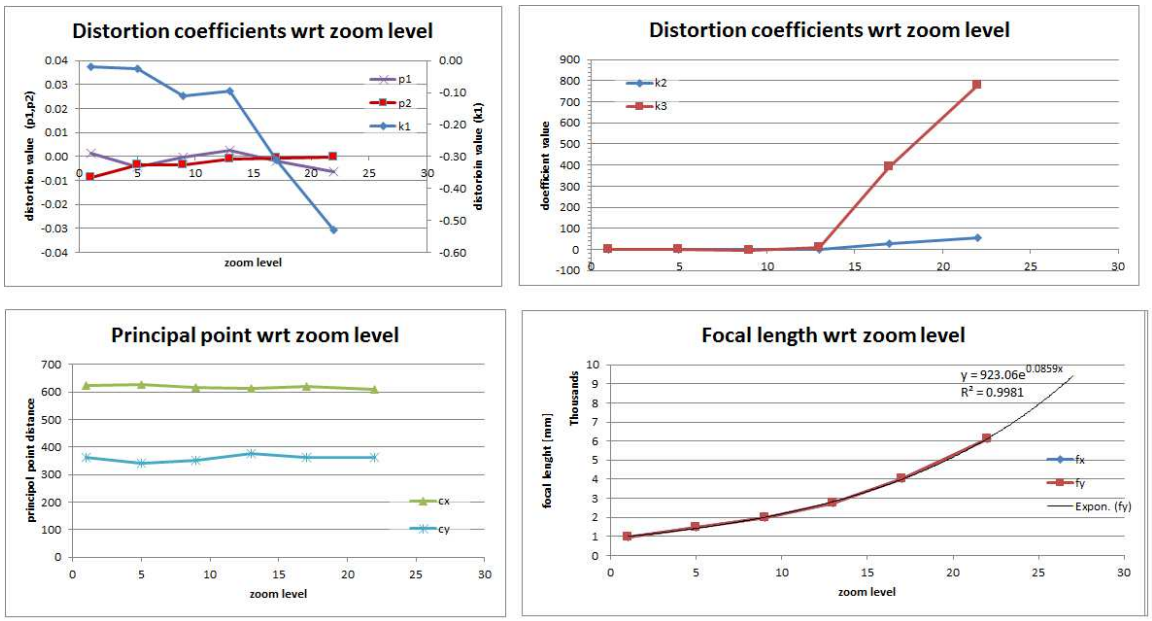

Figure 5.3: The final derived intrinsics for the camera. It shows the improved parameters at zoom level 17, and the exponential fit of the focal length.

5.2 Camera model result

The camera model is divided into two parts, as described in the methodology: there is (1) initialisation of the camera parameters based on GPS and image points correspondences during the offline phase, and (2) the update of camera parameters throughout the video. We discuss the tests and results of both parts individually and consecutively.

5.2.1 Camera initialisation results

The initialisation process was started by time aligning the GPS data and the video. Then, at the start of the video, between frames 800 and 950, iteratively the image correspondence of the GPS locations are selected. The GPS detection refers to the location of the GPS receiver, which is attached to the upper back of the player, and is depicted by a (X, Y) coordinate. We match a GPS detection to the image coordinate of the feet of the respective player in the image plane. This process is performed manually for each GPS detection in a single frame, and repeated over a series of frames.

The GPS data file recorded a world coordinate approximately every 20 ms, although the GPS values only change every 200ms. We only use the first occurrence of a series of repeating measurements. To determine the actual location at any time, positions are interpolated linearly in between data points.

We have matched a total of 142 GPS detections to their respective image locations. The 2D-3D correspondences are used to find the camera intrinsics and extrinsics, solving the PnP problem iteratively, as described in Section 4.2.4. The results are shown in the table below:

Table 5.1: Camera initialisation results.

zoomlevel z	3.24
camera position (X,Y,Z)	-0.48m, -39.1m , 5.49m
camera orientation (around X,Y,Z)	95.9°,-4.72°, $1.4 \cdot 10^{-3}$°
average reprojection error:	15 pixels
number of 2D-3D correspondences	142

The average reprojection error depicted, is the average reprojection error of all the inliers, and has a value of 15 pixels. Due to the state of the field, the lines are hardly visible and worn away, such that corner points are hardly identifiable. This disallows us to use a ground truth based on the geometry of the field. Additionally, the exact field dimensions are also unknown, making it difficult to find the exact cause.

To visualise the error, we create a model of the field which we project onto the image plane. This result is shown on the left in Figure 5.4. It can be seen that the roll error is significant, not following the side line accordingly. The pan angle can be seen to be slightly off as well, allowing for an increasing deviation from the drawn mid line to actual mid line as the distance to the camera increases. This offset influences the entire camera model, since it is based on this initialisation and the deduction of the camera movement relative to these parameters. On the right of the figure, the reprojection errors are shown. It can be seen that although there are a few outliers, but generally the error is in the order of 10 to 20 pixels, and the direction is rather consistent for a single person. I the frames that were used for the initialisation, most players weren't moving, explaining the clusters of points.

We deduct that there are two possible errors: (1) the camera calibration has not been performed correctly or densely enough to capture the intrinsic values of the camera throughout the whole zoom range, and (2) the initialisation data is too noisy and/or the time sync is imperfect. The latter is heavily dependent on the accuracy of the GPS data. The GPS has been pre-processed before it was acquired from Johan-sports, such that the GPS coordinates refer to locations with respect to the midpoint of the field. Whilst this process

Figure 5.4: Initial camera parameters. On the left is the projection of the field onto the image plane. On the right, the projected GPS points are connected to the hand-picked player locations. The camera parameters were found by iteratively solving the PnP problem using a Levenberg-Marquardt optimisation.

normally occurs by using GPS detections taken at each of the corners of the field, for this series of matches the field corner locations have been taken from a different source: Google maps. As a result, the given player location data might contain a translational shift and a rotation as well. This makes the GPS data less reliable.

In order to allow the camera model to be used for processing the video, an attempt is made to build on the existing initialisation and improve it by hand. Although this is a tedious process if it has to be performed for each and every video, it will allow for better processing, and the initialisation will not jeopardise the entire algorithm. The results are shown on the in Figure 5.5. The corresponding values are as show in Table 5.2. The average reprojection error gives a result of 30 pixels (twice as much as the original), which strengthens our hypothesis that the GPS data is shifted.

Table 5.2: Camera initialisation results manual fix.

zoomlevel z	5.027
camera position (X,Y,Z)	-0.26m, -40m , 3.2m
camera orientation (around X,Y,Z)	94.0°,-4.41°, 0.055°
average reprojection error:	30.0 pixels
number of 2D-3D correspondences	142

The fix that is performed it not guaranteed to be the perfect initialisation. For the frame that the correction is imposed upon, the backprojection of the field is correct, however, there is no certainty that this is the only solution. Because there are only two lines available in the frame plus a rough estimation of the mid point, against 7 parameters that need to be estimated, there is a multitude of solutions. The solution presented here is a reasonable one, which was visually verified during the manual correction process.

5.2.2 Online parameter update

For the result on the online video processing, we face the same problems as with the initialisation. Because there are hardly any points in the image plane of which the corresponding 3D world location is known, we are not able to use the projection of the field and compare it to the visual points. Although it would be possible to annotate the lines by hand by

Figure 5.5: Initial camera parameters. On the left is the outcome from the player point correspondences, and on the right is the outcome from manual optimisation.

a visual estimation and geometry scan of the frame, this will only allow for line to point comparisons, and not a point by point comparison. This comes with disadvantages as well, since the exact size of the field is not known as well, and the projected field onto the image might not be meant to fit onto the field lines on the image plane. To avoid the above problem and to save resources, we decided to perform a qualitative evaluation of the camera model. For a series of model configurations we visually check the similarity between the field and the projected field from the found camera parameters. During the playback of the video, we record the error function, as well as the individual components that make it up, as defined by Equation 4.6.

After a series of configurations, the weights of the following configuration gave us the best result.

$$\begin{aligned} \text{reprojection error weight } \omega_{rp} &= 1 \\ \text{homography reprojection error weight } \omega_H &= 5 \\ \text{zoom continuity weight } \omega_z &= 1000/2 \\ \text{camera location weight } \omega_T &= 0.5 \end{aligned} \tag{5.1}$$

Because the initialisation of the camera shows flaws, it is believed that a correct homography between the different frames is of a high importance. Since the reprojection error is derived from the matched keypoints, the inliers for which the homography is below the threshold of 3.0 pixels are used to find the error. The zoom error weight is set to a relatively high value, because the error itself is generally very small due to its definition, where the squared zoom difference is divided by previous focal length. Since the focal length is in the order of 10^3, the unweighted error is in the order of 10^{-4} for deviations up to 1 zoom level. To emphasise the importance of the continuity of the zoom level, the weight has been set to 500. Early experiments have shown this weight, in combination with the translation weight to be sufficient to guide the solution towards a continuous zoom level and static location without compromising on the other error parts.

To show the results of the camera model, we show the propagation of the error through time for a short series of frames. In Figure 5.6 this sequence is shown for the different camera parameters. And in Figures 5.7, 5.8 and 5.9 the tracked camera parameters, errors, and used baseframes are shown.

Looking at the error chart, it is immediately noticed that the errors from the location, and the error from of zooming have both small unweighed value. The error function generally increases over time, when camera changes are present, and this increases until a new base-frame is selected. Because the error in the figure is shown relative to the used baseframe, it is expected that upon instantiation of a new base frame the error reduces to at most the error that the new baseframe has with respect to its own baseframe. When looking at the baseframe diagram, it can be readily seen that rotations have a direct influence on the baseframe assignment and update process as expected. By rotating the camera, the overlap between two frames decreases, and thus the area over which the key points of both frames can be matched, decreases as well, until the lower boundary has reached and a new baseframe is appointed. The figure also shows the that zooming has a large influence on appointing new base frames, which is explained by the same argument.

Figure 5.6: An image sequence portraying the projection of the field lines onto them. It shows several successful and unsuccessful projections.

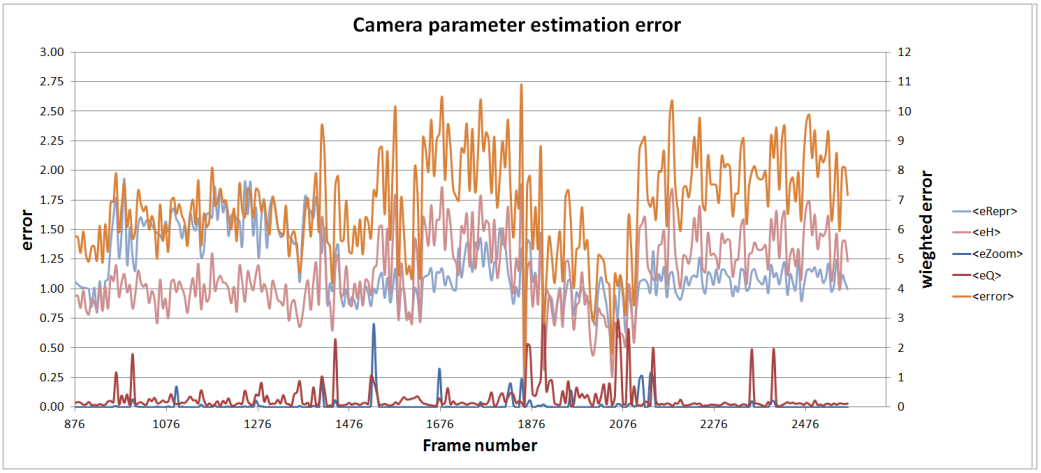

Figure 5.7: Error propagation of the camera for a short sequence. The weighted error are shown relative to the secondary vertical axis on the right, the individual, unweighed errors are shown relative to the left vertical axis.

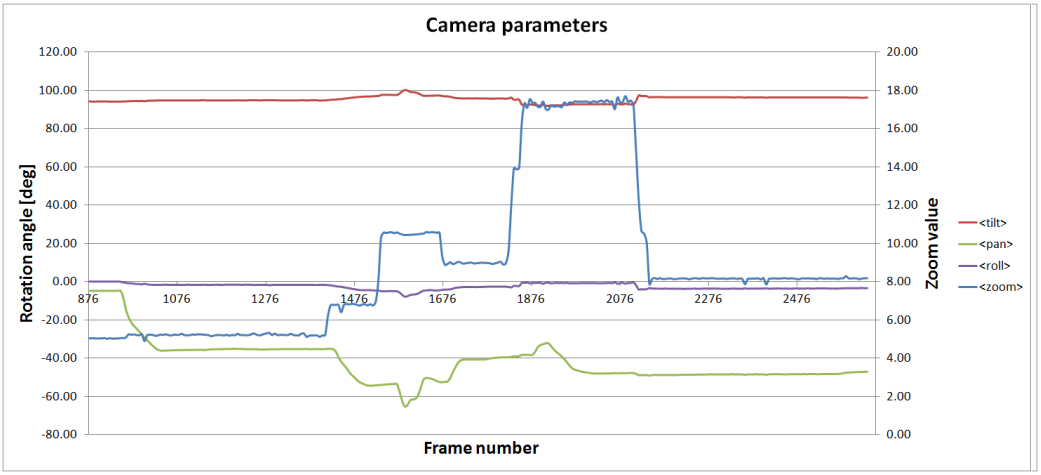

Figure 5.8: Rotation parameters for a short sequence. The rotation parameters are shown with respect to the left vertical axis, the zoom value with respect to the right. Although the model doesn't restrict rotation around the camera look along axis (z-axis), there is hardly any roll present.

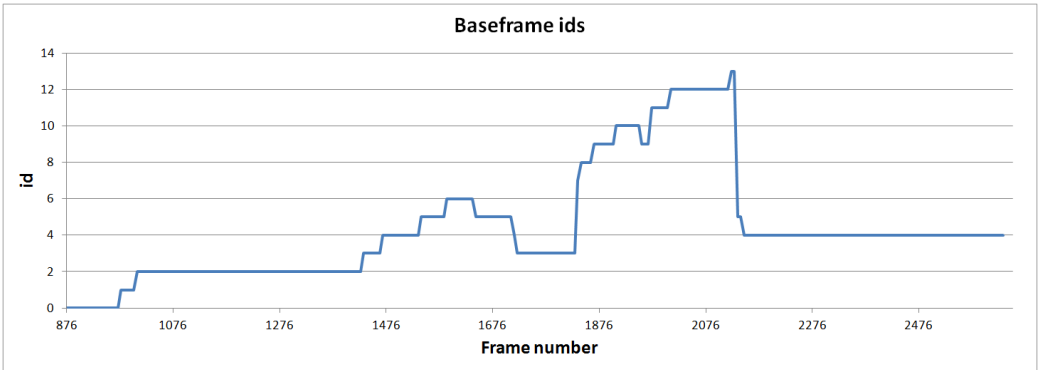

Figure 5.9: Overview of active baseframe, given by id. Baseframe backtracking occurs several times whenever the camera rotates back to an orientation that has occurred in the past.

5.3 Player detection results

The quality of the player detection is tested using the precision and recall for the detections. The detections that are found with the deformable parts model by Felzenszwalb, undergo several steps before being used for the tracking algorithm. These steps are, as explained in Section 4.3:

1. filter detections with background color model
2. filter detections with team colour models
3. filter detections with height and location from camera projection

A detailed analysis of the effects of the different filters was performed on the first two criteria only. This way, the detector is tested independently of the camera model. The analysis compares the performance of the resulting detections in terms of precision and recall. A series of frames throughout the video is manually annotated with ground truth detections. These detections are compared to the outcome of the player detector. We make use of the publicly available pedestrian model generated from the VOC-2010 database. For the ground truth detections, approximately half of all detections have some form of occlusion.

We test our detector with several cases. As a baseline, we use the detector output as-is, after which we add the filters independently. We introduce the background colour model as first, after which we add one of the team colour models. For each of the team colour models, multiple series are compared for different assignment confidence levels. In the case where all histograms are compared, the threshold is the average histogram distance according to the Chi Squared distance. In case of the selective colour descriptor the distance is given by the Mahanalobis distance.

The performance of the player detection is given by the recall-precision graph, as shown in Figure 5.10. The recall and precision is given for a series of confidence levels of the detector model, ranging from -1.3 to 0.5. Recall is defined as:

$$\text{recall} = \frac{\#\text{ of correctly found detection}}{\#\text{ of ground truth detection}} \tag{5.2}$$

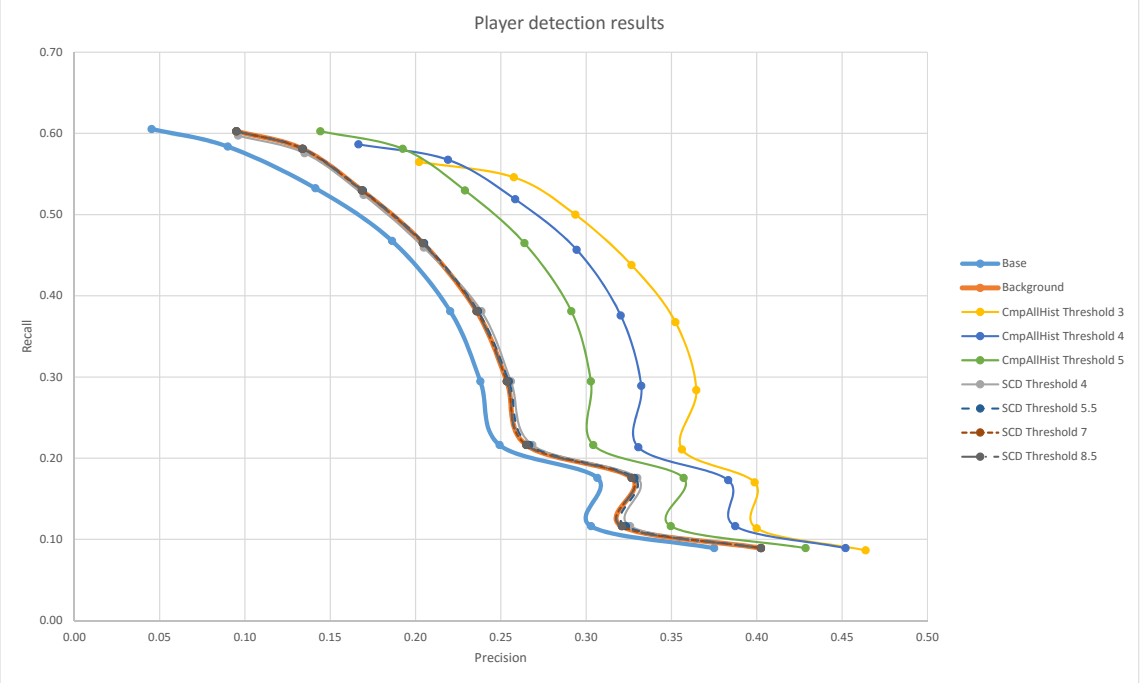

Figure 5.10: Player detections results for the various filters.

and precision is defined as:

$$\text{precision} = \frac{\#\text{ of correctly found detections}}{\#\text{ of found detections}} \tag{5.3}$$

We count a detection as a true positive when at least 50% of their intersecting area is covered by their intersecting part, i.e.

$$\frac{|\text{ detection} \cap \text{ground truth }|}{|\text{ detection} \cup \text{ground truth }|} > 50\% \tag{5.4}$$

Figure 5.10 shows that for the baseline, the maximum achievable recall is 60% at a confidence level of -1.3 of the player detector model. This comes at a very low precision however, which is as low as 5%. The main reason for the low precision results from the fact that there are a lot of supporters around the field, which were detected as well. A visualisation of the spotted detections is given in Figure 5.13, which confirms this. Each row contains the same two frames. The top row contains the result for the baseline. The others are background removal, team assignment using all histograms, and the selective colour descriptor. When considering only the top row, indeed the supporters along the side of the field are being detected. Also, due to the irregularity of the field, areas which are part of the field are returned as a detection as well.

A first measure against rejecting the false detections, which occur mostly in the sky or on the field is by getting rid of detections that are largely considered to be background, based on the background colour model. We have picked a total of 17 colour bins from the HSV

histograms created from a series of images over the entire range. The histogram consists of 15 by 8 bins, ignoring the Value channel for illumination invariance, plus an additional two for white and black. From the field histogram, only considering the bottom 80% of the image, 10 bins were picked, and the other 7 came from the histogram that considered the top 10% of the image, to ensure the blue sky and mountains were contained.

The results of background subtraction show almost no decrease in recall, meaning that only a few of the actual detections are seen as background. In the figure, the line representing the background removal filter is almost completely overlapped by the selective colour descriptor threshold lines. The accuracy increases significantly for all confidence levels, conforming that false detection are actually a detection of the background. There people that stand along the lines are not filtered out, as well as the trees and other background elements. This is shown in the second row of Figure 5.13.

For the two team colour models, each one is added to the background filtered detections independently. The colour model that compares all the histograms is shown for assignment threshold ranging from 2 to 5, and the selective colour descriptor is shown for thresholds between 4 and 8.5. The first notice is the near-identical performance of the selective colour descriptor compared to the background removal. This means that the colour model actually does not filter out bystanders and non-player detections correctly. The comparison with all histograms however does show improvements in precision. Although one would expect the precision to become even larger, the colour model shows it defects. To go more into detail on the team assignment results, individual results have been generated for both teams. These are shown in Figures 5.11 and 5.12.

A big downside to both the selective colour descriptor as well as the comparing histogram descriptor is that these do not incorporate occlusions by other players. If a Dutch player is occluded by an opponent, the colour descriptor or histogram will register the colours of the occluder as well, pulling the decision towards the occluder, instead of the one that is occluded.

The main difference in the results from the team colour methods can be found by acknowledging that the colours of the opponent are black and have dark shades of gray. Although we have added an additional bin to our histogram for shades black, the model is not able to identify the players of the opponent team correctly. This is visible from the opponents detection results, in Figure 5.12. For comparison the base case of the detector has been shown as well. The recall of the team detection is given by the number of correctly assigned detections of that team, divided by the number of ground truth detection of that team. In case a detection gets assigned to the wrong team, it is not included in the results.

Both the full histogram comparison and the selective colour descriptor show over 60% false positives in almost all cases. These false positives come from the crowd, players from the other team and the background itself. It is believed that these results can be improved upon by selecting more histogram bins to be part of the colour descriptor, or by concatenating selected bins together to cumulatively cover a larger regions of the colour space. Team assignment by comparing the entire histogram gives better results than the selective colour descriptor, but the histograms do record parts of the background, since its within the bounding boxes of the detections for training. As a consequence, detections in the background with a similar background to player ratio as the model detections gain an advantage by scoring a high affinity there.

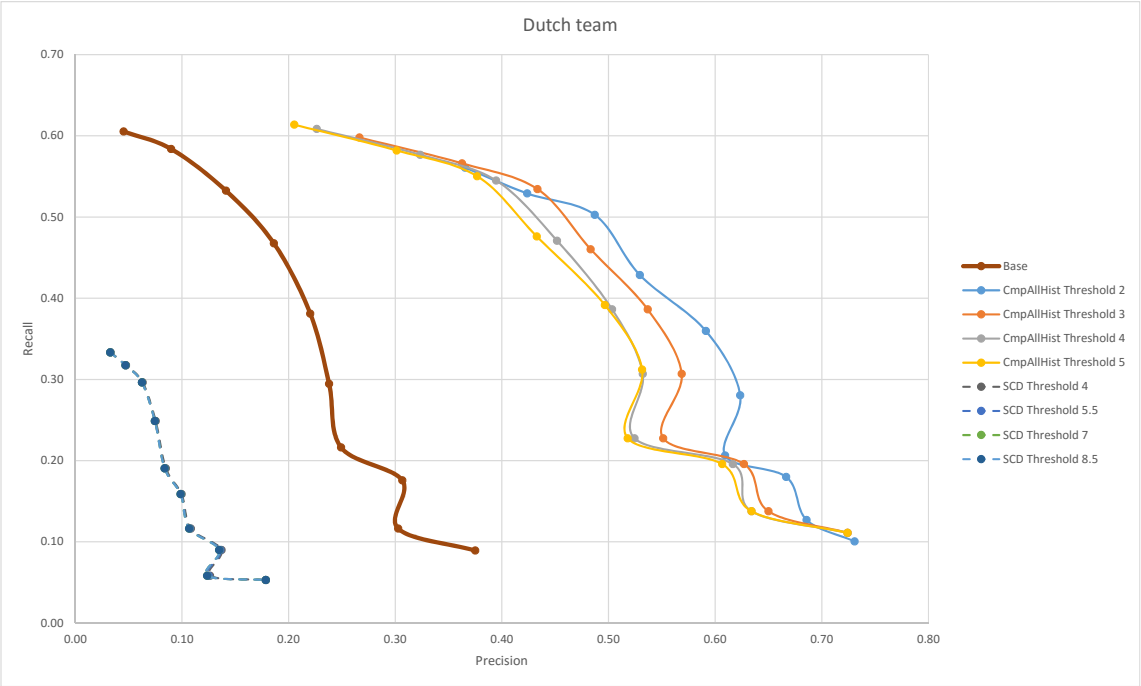

Figure 5.11: Player detections results for the Dutch team.

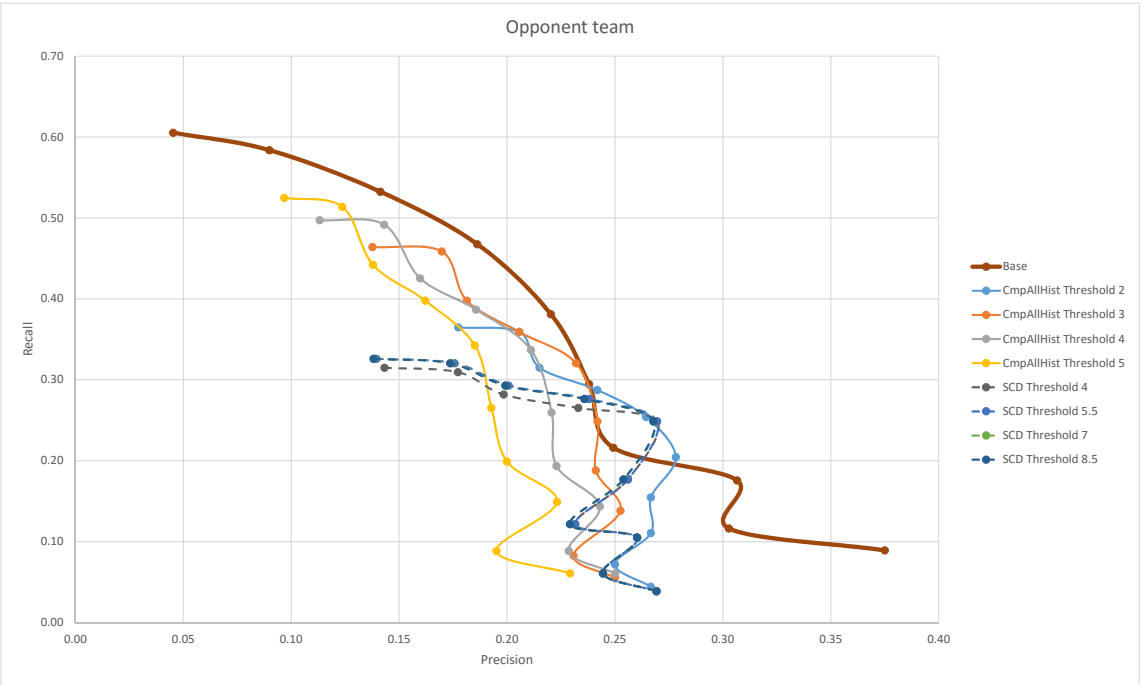

Figure 5.12: Player detections results for the opponent team.

Figure 5.13: Visualisation of player detection results for the different filters. For both frames, the detector confidence level was set at -1.3. From top to bottom: player detector, background removal, assignment using all histograms (threshold=5), and on the bottom the selective colour descriptor (threshold 8.5).

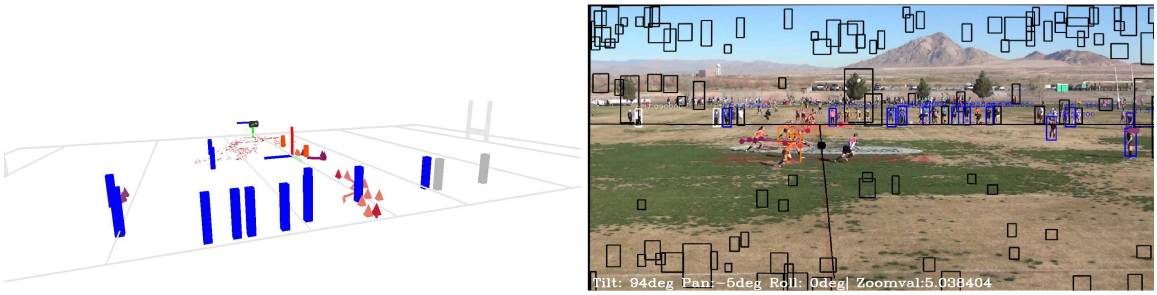

Figure 5.14: Example frame with field lines and detections showing that detections outside of the crowd often still fall within the image. Because of the low team assignment accuracy, people are still detected.

For the remainder of the results, we will use the team assignments based on the comparison with all model histograms with a threshold of 3, at a detection confidence level at -0.9. This setting is a compromise between recall and precision, such that on average 50% of all players are found for the right team at each frame.

5.4 Tracking results

For the tracking results, we intent to test the outcome of our tracking methodology by comparing the results to the given GPS data. The video is split up into smaller parts from kick-off to try, drop goal or penalty kick. As soon as points are scored, the video gets cluttered with extra players which hand out water bottles, but are not participating in the game, and are therefore not to be tracked. Also, the position information between scoring and the next kick-off is irrelevant for any statistical analysis and is therefore left out.

In Figure 5.14, a result of the tracking algorithm is shown very early in the game. On the left, there is shown a 3D model of the rugby field, according to the official size by the rugby federation. On the right the image frame is visible, with the field lines drawn over them. All the rejected detections by the background colour model, projection location, or projection height are shown in black. The remaining detections are team assigned detections, with white being unassigned. What is most striking is the projected location of some of the people along the sideline. Many of the spectator's bounding boxes bottom border is only a 2-4 pixels away from the projected sideline. As a consequence these are indeed projected onto the field. Because the precision of the team classifier is low, the spectators start being tracked. This is something which is noticed continuously throughout the video, and is something that can only be accounted for by increasing the accuracy of the team classifier.

The validation of the tracking algorithm would ideally be performed by comparing the tracklets to the GPS data. Unfortunately, we had to report the GPS location discrepancy with respect to the field's coordinates during the initialisation results. Attempts to compare the tracklets with the GPS data gave back the following results:

The metrics above are defined as follows: we annotate a GPS track as *mostly tracked* when at least 80% of the processed frames, a tracklet was matched to a GPS track. We use a greedy Hungarian matching algorithm to match GPS tracks to tracklets, and allow for matches as long as the distance is within 3 meters. When matching tracks to detections,

Table 5.3: Tracking results for the first kick-off to try sequence. The results are from the comparison between the found tracklets and the GPS detections.

Mostly tracked	0%
Mostly lost	100%
id switches	9
Total nr of tracklets	242
Average assignment error per frame	0.1

no temporal information is used, such as direction or velocity. A GPS track is *mostly lost* if it is not matched to a tracklet for 80% of the processed frames. It shows that no GPS tracked is actually found. A quick visual inspection shows that throughout the sequence, the backprojection of the detections are reasonable, assuming a standard size rugby field. Therefore, we decide to perform a visual inspection on the resulting behaviour

When we perform a visual inspection on the combined camera and tracking behaviour, the following findings come to our attention. The number of resulting tracklets which are active per frame is in most cases higher then the number of actual players on the field. This is because the supporters on the far side of the field are being tracked as well. Occlusions are problematic as well. As soon as a player becomes partially occluded, but is still detected, the colour function spikes, which in turn gives a high total cost for the detection-tracklet combination. As a consequence, the tracklet is not updated, and a new one is initiated. This tracklet continues to exist as long as the occlusion ratio slowly changes, such that the colour histogram of matched detections are equal as well.

It is noticed that a tracklet that was successfully matched to detections a number of consecutive frames, is sometimes unable to be matched to a nearby detection, even when the size and colours are very similar. This results in the creation of a new tracklet, and detections are assigned to the old and new tracklet back and forth. A possible explanation for this behaviour is that with each successful detection match, the variance of the tracklet decreases. This is the property of the Kalman filter that it is most praised for, and allows for very precise prediction. However, if there is no detection match, the prediction for the current frame is used to predict the location in the next, increasing the total error covariance through the control error. Because the backprojections of the detections show large variations, the reduced variance of the earlier successful tracklet's dynamic model is not able to recognise the new detection with deviation anymore within the first few standard deviations. Tests with an increased variance on both the prediction as well as the detection show that the behaviour wears off. But, as a downside, due to missing detections, id switches between tracklets start increasing. And, since there are still many missing detections, the few detections available are all assigned to only a handful of tracklets, who's position history starts looking more and more erratic. Two examples of back and forth going tracklets are shown in Figure 5.16.

Another important behavioural property is the following: If two tracklets get close, and in the next frame, only one person is discovered by the detector, the cost function is not specific enough to ensure the assignment of the detection to the correct tracklet. As a consequence, one of the two tracklets gets assigned the detection, while the other tracklet's variance increases. In the next frame, if only one detection is found again, the tracklet with the lower variance is assigned the detection. This process repeats until the player separate

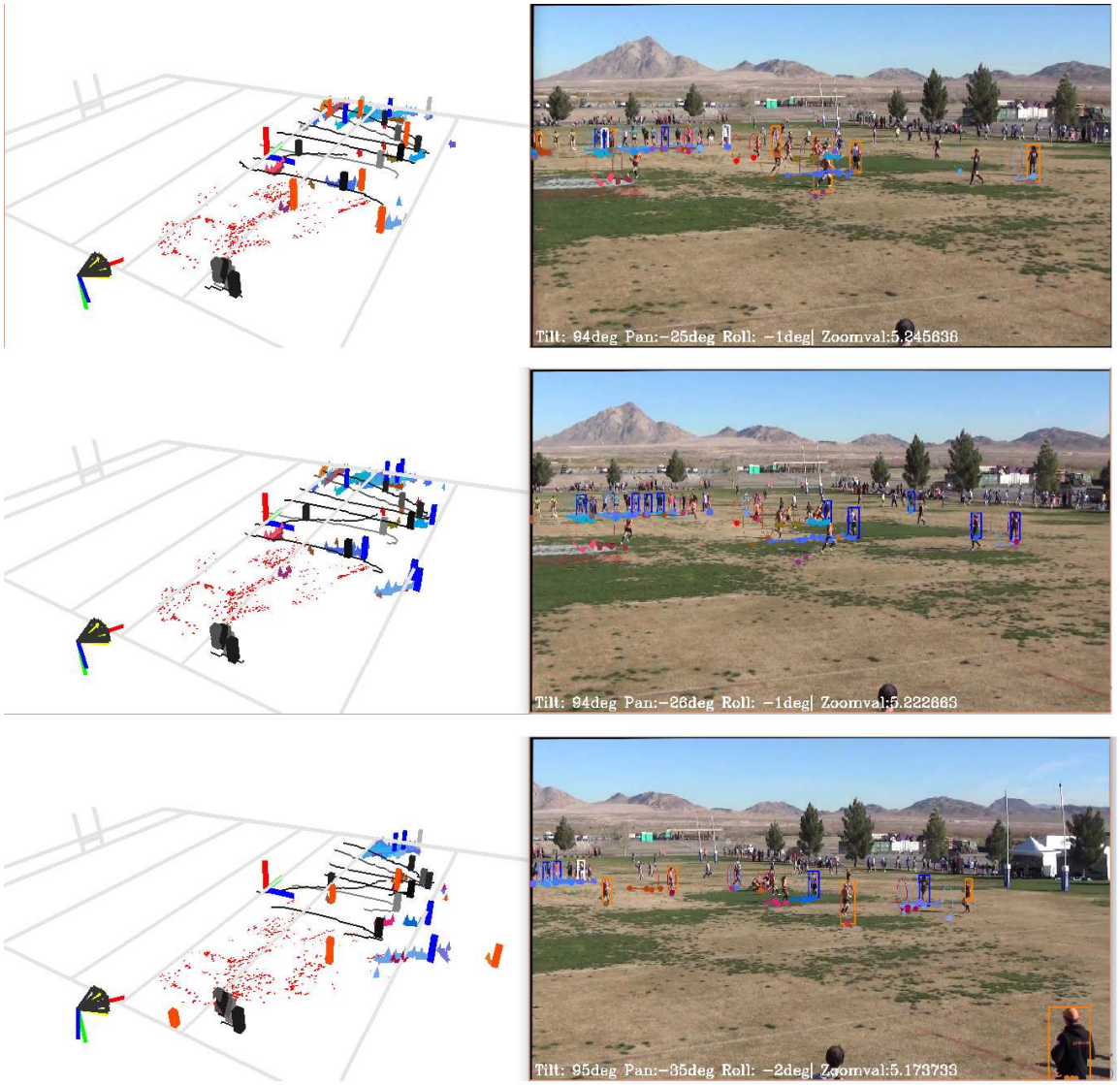

Figure 5.15: Examples of tracklet updates during tracking. The black and gray lines colours refer to GPS tracks and current detections. The blue and purple tinted lines are of the opponent team, and the orange/green/pink colours lines to the Dutch team.

Figure 5.16: Consecutive detections being assigned to two overlapping tracklets. On the left, the progress is shown in three frames. On the right an example is given, where the ground plane path history of the tracklets show a similar situation.

and go their own way, but there is no guarantee that the tracklets follow the correct player. Thus, during the part where the players are close, many id switches occur. And when the players split again another one possibly happens. This happens more often with players which are further away from the camera, as the assigned variance for the detections grows with increasing distance from the camera.

5.5 Discussion

We relate back our results to our research question and objectives. In our research question we stated: *"to what extent is it possible to give 2D ground plane location data (...) of all players (...)"*. The research question was split up into three subquestions: (1) What are the camera parameters throughout the video, (2) How precise and accurate can players automatically be detected in video frames, and (3) how can players be tracked within the video. We discuss the outcome of each of them.

For the assessment of the first subquestions, determining the camera parameters through the video, we have come up with a methodology that allows for variation in zoom level. It is heavily dependent on the initialisation of the camera parameters, which are determined before the online processing. During the playback of the video, iteratively the new parameters are found, which relate back to a reference frame. This methodology does not fix the camera location, nor the roll parameter, although we discard the newly found camera location and stick to the one from initialisation. As a consequence, reprojection errors occur and grow over time. The results of the reprojection projected field lines on the image plane, show that the camera parameters are nearly correct for camera states close to the initial one. However, with increasing deviations from this initial state, the errors grow along. In Section 4.1.1 we state that we assume that there is a continuity constraint on the camera parameters, however, these constraints are not exploited by the model.

The precision and accuracy of player detection has been shown in the detection, where player are detected with a recall up to 60%. The precision parameter, using the detector without any post processing shows a low precision of less than 4%, meaning there are many false positives. Increasing the confidence threshold of the detector, reduces the amount of false positives, but at a cost of the recall. We have used a number of methods to

reduce the number of false positives. By discarding detections of which the majority of the pixels in the bounding box have been assigned as background, the accuracy improves without compromising on recall. However, for the team assignment improvement, many false assignments are present, causing a reduction in recall and precision. As a consequence we have to state that the performance of the team classifier is poor.

The tracking algorithm builds heavily on the results of the first two subquestions. In our methodology, we presented a dynamic model, the Kalman filter, to assist in the matching of detections throughout the consecutive frames. Because at each frame, on average 50% of the players are undetected, the tracking behaviour, the dynamic model holds a large uncertainty allowing for detections of nearby players to be selected. The results of the tracking algorithm show that many tracklets are being created, which all last relatively short. As a consequence, there are no complete player tracklets generated, but tracklets of generally 2 to 5 seconds.

There are a number of desirable changes that could readily improve our methodologies' performance and, more importantly, allow for better and more quantitative testing:

- More distinctive team colour model by means of a (linear) classifier.
- Having the exact measurements of the field available.
- GPS data for which the corners of the field are measured with the same device, to lose precision problems, and allow for a more elaborate and quantitative analysis of the tracking algorithm.
- Known zoom value at start of the video, or a series of reference points with known location.
- Different position of the camera with respect to the field. Preferably up such that the angle with respect to the horizon is at least 20 degrees. This will allow for easier filtering of background, will reduce occlusions and allow for a more robust camera parameter estimation and detection backprojection.
- Video recording on a different field, such that the field lines are visible, allowing them to serve as a reference. This is mainly for validation purposes.

Chapter 6

Conclusion

The goal of this project was to present a methodology that allows for robust tracking of rugby players from video material. We have presented our methodology, which had three distinct part: (1) the derivation of the camera parameters stating the camera's location, orientation and internal settings such as focal length, (2) the detection of players throughout the the video, and (3) the association of detections and tracklets of rugby player locations from the image plane, translated and projected onto the ground plane.

Our methodology shown good initial results for the camera model. Although for the given data the camera initialisation showed its defaults, which were caused by erroneous GPS data, the qualitative evaluation shows promising results. A big advantage of the methodology presented is its ability to recover from reprojection errors acquired earlier during the video sequence, due to the use of baseframes.

The player detections, found by the deformable parts detector of Felzenszwalb et al. [11], were further processed by a series of filters, based on colour and its backprojection to the ground plane. Problems occurred with the team assignment filters, of which the team classification is rather poor. Tying all the previous pieces together, the tracking part of the methodology combined the information of the camera model and the detections for consecutive frames, and created a series of tracklets, which are governed by a dynamic model given by the Kalman filter. The results of the tracklets were difficult to validate, since no ground truth data was available. A qualitative test on the general behaviour of the tracklets showed that the complexity of the system was high due to the severe conditions.

Although our goal has been to create a robust methodology, the results show their defaults. Due to several external factors, such as camera position, bad field state and unknown field size, tracking has become significantly more difficult. As a consequence, much of

the evaluations of our work had to be done on a qualitative basis, whereas a quantitative analysis would allow for specific insights into the performance of the presented system.

6.1 Future work

During the challenge of determining ground plane location data, focus was put on the creation of a continuous camera model. Due to this focus, less resources have been put in the detection and tracking processes. Nevertheless, there are still improvements possible. Currently, during a new baseframe selection, each available baseframe is compared to the new candidate base frame, by finding the homography and the overlap between the candidate and a baseframe. However, using the rotation parameters as a first measure, significant time can be saved by discarding those base frames which will most likely not overlap with the candidate. In a similar way, finding the new camera parameters can be executed more effectively. Since the roll parameter is fixed, as well as the camera location, additional constraints can be added to the PnP solver. This way, the pose estimation problem problem is reduced to finding only 3 parameters instead of 6. Because we deal with distortion, the process is still a iterative one, both at a single zoom level, as well as for finding the optimal zoom level.

To improve the overall performance and to allow for better handling of the severe external conditions, a set-up of multiple camera's would help. Players which are occluded in one camera's field of view, might be discoverable in the next. The location of players who are detected by multiple camera's can be determined more precise by combining the reprojection information. Additionally, a set up with multiple camera's reduces the need for camera rotation and zooming, which in its turn allows for a simpler and more robust camera parameter output.

To boost the performance of the detection model the pre-trained model of Felzenszwalb should be replaced with a model trained specifically for sports players, or specifically for rugby players. This should include a number of components which specify a combination of different poses as well as viewpoints. Especially if poses and orientations that happen when performing tackles and scrums are added to the model, it is believed the performance of the detector will increase significantly.

There is much to be gained for the team classification. A simpler solution as using Gaussian weighted histograms for the detections, or splitting the detections horizontally in two halves can already improve the current model greatly. More sophisticated methods, such as the one used by Lu et al. [21, 11], use a classifier such as the Logistics Regression classifier or the SVM classifier. This allows not only to make better distinctions between teams, but also to classify detections of background clutter correctly.

The tracking methodologies proposed in Section 4.3.5 have the potential to be exploited in a more efficient manner. By including methods from the detection-by-tracking studies, such as particle filters or a meanshift tracker, will allow for location updates of the tracklets in case of no detection match not only based on the dynamic model, but also on the local search results of those methods. This adds location predictions based on appearance tracking. Breitenstein et al. [3] considers a Bayesian dynamic model for each tracklet, which is based on the outcome of both a particle filter and the matched detections. After the online phase,

a post-processing step could be added as well, to allow existing tracklets to be combined, and grow from the small series of 2 - 5 seconds, into tracklets of minutes or even a complete match.

Allowing for more sports related properties, the game context features, as proposed by Jinchen et al.[18], show shows great to use sports related interaction forces. Such features can incorporate for example how players act after they perform a tackle would be a very interesting feature to incorporate into that algorithm.

References

[1] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[2] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000.

[3] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1515–1522. IEEE, 2009.

[4] M. Bujnak, Z. Kukelova, and T. Pajdla. New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion. In *Computer Vision–ACCV 2010*, pages 11–24. Springer, 2011.

[5] W. Choi and S. Savarese. Multiple target tracking in world coordinate with single, minimally calibrated camera. In *Computer Vision–ECCV 2010*, pages 553–567. Springer, 2010.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[7] A. Dearden, Y. Demiris, and O. Grau. Tracking football player movement from a single moving camera using particle filters. In *Proceedings of the 3rd European Conference on Visual Media Production (CVMP), London*, pages 29–37, 2006.

[8] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC*, volume 2, page 7. Citeseer, 2010.

[9] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(4):743–761, 2012.

[10] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010.

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[14] Johan Sports. Gps dataset rugby7s las vegas 2015. Unpublished raw data, 2015.

[15] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[16] D. Lay, S. Lay, and J. McDonald. *Linear Algebra and Its Applications*. Pearson Education, 2014.

[17] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[18] J. Liu, P. Carr, R. T. Collins, and Y. Liu. Tracking sports players with context-conditioned motion models. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1830–1837. IEEE, 2013.

[19] D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[21] W.-L. Lu, J.-A. Ting, K. P. Murphy, and J. J. Little. Identifying players in broadcast sports videos using conditional random fields. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3249–3256. IEEE, 2011.

[22] T. Mathworks. Gaussian mixture parameter estimates - matlab - mathworks benelux, 2015. [Online; accessed 20-August-2015].

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[24] S. N. Sinha and M. Pollefeys. Pan–tilt–zoom camera calibration and high-resolution mosaic generation. *Computer Vision and Image Understanding*, 103(3):170–183, 2006.

[25] C. C. Slama, C. Theurer, S. W. Henriksen, et al. *Manual of photogrammetry.* Number Ed. 4. American Society of photogrammetry, 1980.

[26] P. Sturm. Self-calibration of a moving zoom-lens camera by pre-calibration. *Image and Vision Computing*, 15(8):583–589, 1997.

[27] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer, 2010.

[28] T. Taketomi, K. Okada, G. Yamamoto, J. Miyazaki, and H. Kato. Camera pose estimation under dynamic intrinsic parameter change for augmented reality. *Computers & Graphics*, 44:11–19, 2014.

[29] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596, 2010.

[30] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[31] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 32–39. IEEE, 2009.

[32] G. Welch and G. Bishop. Course 8–an introduction to the kalman filter. *SIGGRAPH 2001 Courses*, 2001.

[33] R. G. Willson. Modeling and calibration of automated zoom lenses. In *Photonics for Industrial Applications*, pages 170–186. International Society for Optics and Photonics, 1994.

[34] B. Wu and R. Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[35] Z. Wu, J. Zhang, and M. Betke. Online motion agreement tracking. In *Proc. BMVC*, 2013.

[36] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[37] C. Yu and G. Sharma. Plane-based calibration of cameras with zoom variation. In *Proceedings of SPIE*, volume 6077, pages 352–360, 2006.

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

[39] G. Zhu, C. Xu, Q. Huang, and W. Gao. Automatic multi-player detection and tracking in broadcast sports video using support vector machine and particle filter. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 1629–1632. IEEE, 2006.