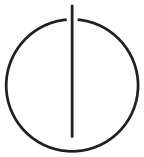
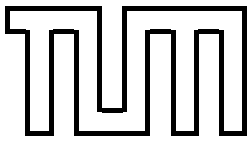


TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik



Continual and Robust Estimation of Camera Parameters in Broadcasted Sports Games

Dissertation

Suat Gedikli

TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik

Continual and Robust Estimation of Camera Parameters in Broadcasted Sports Games

Suat Gedikli

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. A. Knoll

Prüfer der Dissertation:

1. Univ.-Prof. Dr. B. Radig
2. Univ.-Prof. Dr. D. Burschka

Die Dissertation wurde am 22.04.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05.03.2009 angenommen.

Abstract

Given the ubiquity of camera systems, there is an ever growing need for the automatic interpretation of unannotated video recordings. Among the various contents of video streams, actions are of course the most interesting. They represent spatio-temporal events that cannot be spatially interpreted without additional knowledge about the respective camera system, since information about perspective is not directly encoded in the video images. Using domain knowledge, however, it is possible to reconstruct the mapping characteristics of a camera from a recorded image sequence alone. This dissertation investigates the automatic estimation of camera parameters in television broadcasts of sports games played on courts or fields.

We present a system that is capable of automatically initializing and tracking the camera parameters of rotating and zooming cameras. An initial calibration of constant parameters of a particular camera of interest is supported using minimal manual effort. The system can then identify sequences recorded by this camera and differentiate them from slow-motion replays or intercuts of other cameras. The reinitialization after cuts or cross-fades is performed automatically using visible field lines. The dynamic camera parameters of the sequence that follows are then continuously determined through optimization techniques, drawing upon motion estimation and field line detection. Our approach is characterized by a high degree of robustness and efficiency.

We empirically evaluated the reliability of the system by applying it to several full-length recordings of soccer games, which were mainly taken from the FIFA World Cup 2006 in Germany. The system was able to correctly track the main camera from the final game throughout a full 90 minutes without manual intervention. The information collected by the system presented in this work builds the very foundation of the comprehensive sports analysis software ASPOGAMO , which illustrates the practical applicability of the system.

Kurzfassung

Mit steigendem Einsatz von Kamerasystemen wächst der Bedarf nach automatischer Interpretation von nicht annotierten Videoaufzeichnungen. Aktionen als interessante Inhalte von Videos repräsentieren spatio-temporale Zusammenhänge und sind ohne zusätzliches Wissen über die Kamera räumlich nicht interpretierbar, da im Videobild die Perspektive der aufzeichnenden Kamera nicht kodiert ist. Mit Hilfe von Hintergrundwissen können die Abbildungseigenschaften einer Kamera allein aus aufgezeichneten Bildfolgen rekonstruiert werden. Diese Dissertation untersucht die automatische Schätzung von Kameraparametern in Fernsehübertragungen von Feldsportarten.

Wir stellen ein System vor, das in der Lage ist, die Kameraparameter von schwenkbaren Fernsehkameras mit Zoom aus Sportübertragungen automatisch zu initialisieren und nachzuführen. Das System ermöglicht die Kalibrierung konstanter Parameter einer Kamera mit minimalem manuellen Aufwand. Videosequenzen von dieser Kamera werden zwischen Zeitlupenwiederholung und Aufnahmen anderer Kameras im Videostrom identifiziert. Die Initialisierung der Kameraparameter nach Schnitten oder Überblendungen erfolgt automatisch anhand sichtbarer Feldlinien. Die dynamischen Kameraparameter des folgenden Videosegments werden kontinuierlich mit Hilfe von Optimierungsverfahren aus Bewegungsschätzung und Feldlinienerkennung ermittelt. Das Verfahren zeichnet sich durch hohe Robustheit und Effizienz aus.

Die Zuverlässigkeit des Systems wurde empirisch an Aufzeichnungen von Fußballspielen, hauptsächlich von der FIFA Weltmeisterschaft 2006 in Deutschland, evaluiert. Das System war in der Lage ohne manuelle Eingriffe die Hauptkamera der Fernsehübertragung über 90 Minuten des Finalspiels hinweg zu verfolgen. Die gewonnenen Informationen aus der praktischen Anwendung bilden die Grundlage für Sportanalyse im ASPOGAMO -System.

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Bildverstehen und Wissensbasierte Systeme der Fakultät für Informatik der Technischen Universität München.

Zuerst bedanke ich mich bei meinem Doktorvater Prof. Dr. Bernd Radig für die Aufgabenstellung, seine Unterstützung und das in mich gesetzte Vertrauen. Bedanken möchte ich mich auch bei Prof. Dr. Darius Burschka, der sich bereit erklärt hat, als Zweitgutachter zu fungieren.

Mein besonderer Dank gilt Prof. Michael Beetz, PhD, für seine Betreuung, Motivation und dafür, daß er mir die Möglichkeit gegeben hat, diese interessante Aufgabenstellung innerhalb des Projektes ASPOGAMO zu bearbeiten. Er sowie Prof. Radig verstanden es stets für ein angenehmes Arbeitsklima zu sorgen.

Ich danke auch meinen Kollegen Jan Bandouch, Zahid Riaz, Dominik Jain, Radu Bogdan Rusu, Dejan Pangercic und Francisco Siles, die Teile dieser Arbeit in sprachlicher Hinsicht durchgesehen haben und somit zur besseren Verständlichkeit beigetragen haben. Meinen Kollegen Nicolai v. Huene und Bernhard Kirchlechner danke ich für die Überprüfung der meisten mathematischen Herleitungen und Formeln in dieser Arbeit.

Außerdem bedanke ich mich bei meinen Kollegen Freek Stulp, Heiko Gottschling, Matthias Wimmer, Simone Hämmerle, Thorsten Schmitt, Murat Duruş und allen anderen für die angenehme Arbeitsatmosphäre am Lehrstuhl und für die sehr interessanten und auch nicht immer fachlichen Diskussionen.

Meiner Familie, insbesondere meinem Vater und meiner Mutter, danke ich für ihre Unterstützung in jeglicher Hinsicht.

Schließlich danke ich meiner Ehefrau Semra, die mit mir alle Höhen und Tiefen meiner Promotion durchlaufen und mir sehr viel Verständnis entgegengebracht hat. Sie gab mir viel Kraft und war mein Fels in der Brandung. Ich hoffe, ich kann die vielen versäumten Stunden mit ihr wieder gut machen.

München, April 2008

Contents

List of Figures

List of Tables

Glossary of Notation

SYMBOL	MEANING
$.^T$	superscript denoting vector / matrix transpose.
A^{-1}	superscript denoting matrix inverse.
$\mathrm{tr}\,(A)$	trace of matrix A.
$\det(A)$	determinant of matrix A.
$\mathrm{rank}\,(A)$	row/column rank of matrix A.
$\mathrm{diag}\,(a_1, \ldots, a_n)$	$n \times n$ diagonal matrix with the diagonal elements $a_1, \ldots, a_n$.
I_n	identity matrix of size $n \times n$.
$\mathbf{0}$	zero vector.
$\breve{\boldsymbol{p}}$	point in Euclidean space.
$\boldsymbol{p}$	homogeneous point in projective space.
R_x, R_y, R_z	rotation matrices around the x, y and z axis respectively.
λ	scaling factor.
$\mathbb{R}^3$	Euclidean space.
$\mathbb{P}^3$	projective space.

SYMBOL	MEANING
f	focal length of the pinhole camera model.
f_x, f_y	focal lengths in units of the pixel width and pixel height respectively.
C_x, C_y	principal point - coordinates of center of radial distortion.
S_x, S_y	metric size of a sensor element (pixel).
κ	radial distortion coefficient.
s	skew - describes the rectangularity of the sensor elements (pixels).
α	tilt angle of the camera.
β	roll angle of the camera.
γ	pan angle of the camera.
$\breve{R}_c$	rotation matrix of the camera describing its orientation.
$\breve{\boldsymbol{t}}_c$	translation vector of the camera representing its position.
K	camera matrix.
H	homography.
ω	image of the absolute conic (IAC).
Rt	rigid transformation matrix.
$\boldsymbol{x}$	state or parameter vector of the system state.
$\hat{\boldsymbol{x}}$	an estimate of the state vector.

SYMBOL	MEANING
$\Sigma_{\hat{x}\hat{x}}, \hat{P}$	covariance matrix of the estimated state vector.
$\tilde{\boldsymbol{x}}$	a prediction of the state vector.
$\Sigma_{\tilde{x}\tilde{x}}, \tilde{P}$	covariance matrix of the predicted state vector.
$\boldsymbol{x}_t$	state vector $\boldsymbol{x}$ at time t.
$\bar{\boldsymbol{x}}$	mean or expected value of the quantity $\boldsymbol{x}$.
Σ_{xy}	covariance matrix of the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
σ_x^2	variance of the quantity x.
$N(\bar{x}, \sigma_x^2)$	univariate Gaussian normal distribution.
$N(\bar{\boldsymbol{x}}, \Sigma_{xx})$	multivariate Gaussian normal distribution.
$\boldsymbol{x} \sim N(\bar{\boldsymbol{x}}, \Sigma_{xx})$	normally distributed random variable $\boldsymbol{x}$.
$p(\boldsymbol{x})$	probability density function.
$Bel\,(.)$	belief state.
$E[\cdot]$	expectation value.
χ_n^2	chi square distribution with n degrees of freedom.
d_M	Mahalanobis distance.
$\boldsymbol{z}$	observation vector.
$\boldsymbol{r}$	residual vector.
$\boldsymbol{u}$	control vector.
$\boldsymbol{z}_{t_1:t_2}$	set of observations $\boldsymbol{z}_t$ with $t_1 \leq t \leq t2$.
$\boldsymbol{w}$	observation noise.
$\boldsymbol{v}$	process noise.

SYMBOL	MEANING
$\Sigma_{zz}, \Sigma_{ww}, Q$	covariance matrix of the observation noise.
Σ_{vv}, R	covariance matrix of the process noise.
$g(\cdot)$	motion model or system process.
$h(\cdot)$	observation model.
ν	constant factor.
ξ	point tracking rate.
$f(\boldsymbol{x})$	cost function.
J_f	Jacobian of the cost function f with respect to $\boldsymbol{x}$.
∇f	gradient of the cost function f with respect to $\boldsymbol{x}$.
$\boldsymbol{x}^+$	global minimizer of the cost function f.
$\boldsymbol{x}^*$	local minimizer of the cost function f.
Δ_k	trust region radius for iteration k.
l_k	direction for iteration k.
ρ_k	step size for iteration k.
W	weight matrix.
$w(\cdot)$	weight function.

Chapter 1 Introduction

With the availability of huge amounts of video material, e.g. in the world wide web, there is a growing need for the automatic understanding of video content, in particular actions of people. Understanding the actions of people, that are performed at different places, often requires the recognition of their movements relative to the environment they are acting in. Inferring the movement of people and their positions in the environment from video streams requires the knowledge about the mapping characteristics of the recording camera. In this dissertation we investigate this computational problem in a particular application domain: the visual interpretation of sports games played on courts or fields.

We present a system that is capable of automatically initializing and tracking the camera parameters of dynamic cameras from sports broadcasts, particularly of soccer games. We show how such a tracking system can be fully initialized with minimal human help and without access to the recording cameras. Furthermore, the presented system can automatically process video streams taken from TV broadcasts in spite of interruptions of the stream by scenes of other cameras, slow-motion replays, advertisements and special effects like cross-fading and chroma-keying. Scenes that enables the system to estimate player positions are automatically identified, initialized and tracked for usually as long as they last. The robustness and reliability of the system has been extensively evaluated on several full-length recordings of soccer games, mainly taken from World Cup 2006 in Germany [BBG+06, BGB+07, GBvHH+07, BBG+08]. This and the near real-time performance of our camera tracking show the general applicability of the proposed approach to real world applications.

1.1 Motivation

The automatic observation of team sports is gaining increased attention for several reasons. As most sports games, and soccer in particular, are watched by millions, it is a highly competitive and promising market for broadcasting companies. In order to gain new audiences,

they are continuously searching for ways of improving the viewers' experience. For instance, during the last FIFA World Cup 2006 in Germany, highlights and crucial scenes were virtually reenacted after the game, accompanied by expert discussions and analyses. Unprecedented visualizations, particularly for critical scenes, gave the spectators a deep insight into the game and a better understanding of questionable decisions of the referees.

Animations of selected scenes, e.g. goals, from arbitrary virtual camera positions were made available on the Internet a few hours after the game. Such reenactions are done mostly manually and require a massive amount of human intervention for determining the 3D positions of the players and the ball. An automatic estimation of the required data would drastically speed up the process while cutting down costs. The acquired data could also be used to save bandwidth when broadcasting games to mobile devices, where scenes could be animated using game-like computer graphics.

Coaches and major sports clubs are interested in statistics and analyses of their own team as well as opposing teams. A detailed representation of games in the form of a game model enables the study of team-mate interaction and tactics and helps to reveal weaknesses. Teams could improve their interplay and adapt their tactics to the opposing team beforehand. So far, game models are obtained from data gathered by human scouts and therefore are inaccurate, incomplete and subjective. A system gathering the required data automatically could increase the quality of the game models as well as enhance the model with respect to different aspects. Furthermore, this information could be provided by such a system almost in real-time.

Abstract and symbolic representations of games allow sports as well as computer scientists to analyze games and human activities in unprecedented ways. For instance, the huge amount of position data representing real-world multi-agent activities, which lacks the artificial patterns usually found in simulated data, provides interesting opportunities for AI research.

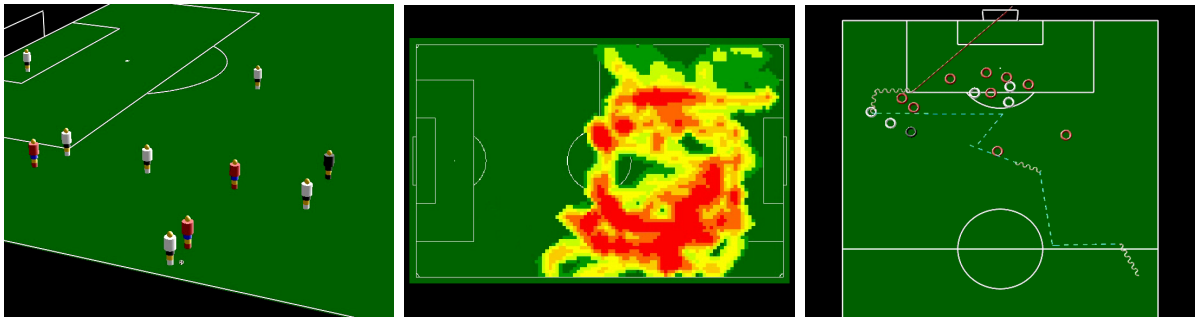


Figure 1.1 Example Applications: Virtual reenaction (left), Statistical Analysis (center) and Tactical Analysis (right) [GBvHH+07]

One of the major design decisions when planning an automatic system for the observation of team sports games is the choice of sensory input (see [Set03]). We will now discuss the available options, examine the benefits and drawbacks and motivate the choices made that have led to this thesis.

Choice of Sensory Input

The automated interpretation of soccer games requires knowledge about the positions of the players and the ball. However, this data can be obtained using different techniques and sensing devices.

One approach followed by the companies CAIROS® and TRAKUS® [Cai, Tra] uses small radio beacons (HF transmitters) mounted on the objects to be tracked in combination with several receiving stations around the field. These systems provide highly accurate measurements at a high time resolution as well as complete datasets for all objects. Nevertheless, the downside is that such systems are invasive and very expensive in hardware, setup and operation. Restricting these systems to a single stadium would reduce the costs in setup, but the utility of such a stationary system is very low. Furthermore, the international football association FIFA (Fédération Internationale de Football Association) has decided on multiple occasions not to allow such invasive systems during official games.

An alternative non-intrusive approach is to use cameras, as they are passive sensors. The common approach is to use multiple static cameras for the observation [FLB+04, IS03, KCM03, THF96, XOJ04]. Such systems have the huge benefit that camera parameters stay constant during the whole game, as does the mapping between image and real world coordinates. In particular, player and ball segmentation can be reduced to an efficient background subtraction and the data association problem can be easily solved in the image plane for most cases. Using an adequate number of cameras, this approach can achieve high accuracy and can provide almost complete observations. However, even if such systems are non-invasive, they are still expensive in hardware, setup and operation. A stationary system bound to a single stadium is unacceptable for the same reasons as mentioned above.

The approach proposed in this thesis does not share the downsides of the systems mentioned above. First, we directly use the dynamic cameras that are being installed by broadcasting companies for live coverage of the games on television. Such cameras are available in any stadium during all relevant national and international games. However, access to the video streams or to the recordings is restricted. Therefore, we go even further by proposing to directly use the video streams from TV broadcasts as they are easily obtainable for everyone

without the need of additional infrastructure. Despite the many difficulties that arise with this method, several reasons make it more attractive than the methods mentioned above. Live video streams of all important games can be received almost everywhere for free. Almost no hardware costs arise, as high-quality semi-professional video equipment is available at low prices. Another crucial advantage is that a vast number of old important games (e.g. past World Cups) is only available on video tapings from TV-broadcasts. Using the proposed method, this pool of interesting historical data can be made available for interpretation and statistical analysis. For example, the DFL (Deutsche Fußball Liga) digitizes all games from the german league from the last 40 years; all in all more than 24000 hours of playing time. Using the proposed system could reduce the costs while increasing the quality of the recovered data.
Figure 1.2 shows an example of an old but very famous game.

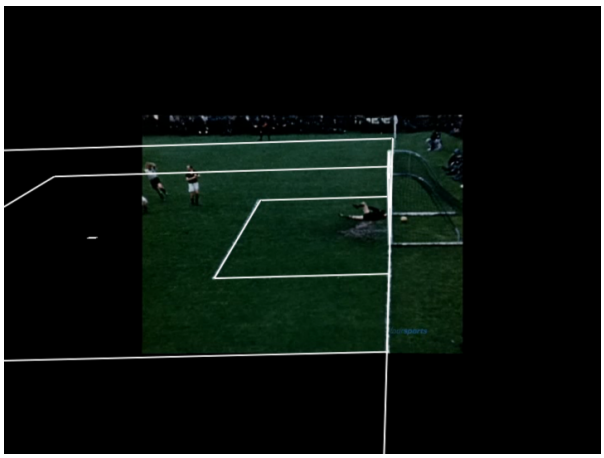


Figure 1.2 Camera Tracking Example: Final of World Cup 1954 in Bern (Switzerland)

Given these considerations, we believe that the proposed method is the most generally applicable solution for most applications, as it can recognize and process video streams from rotating and zooming cameras.

1.2 Problem Description

We will first describe the desired functionality of the proposed system using a specific example taken from the broadcast of the final game of the FIFA World Cup 2006, and then we will point out the problems which have to be overcome by the system.

Figure 1.3 depicts the processing from the 10th to the 12th minute of the game. The video stream contains several video segments that are taken from different perspectives (cameras). They are represented by their first frames in the film strip. The system first has to find the video

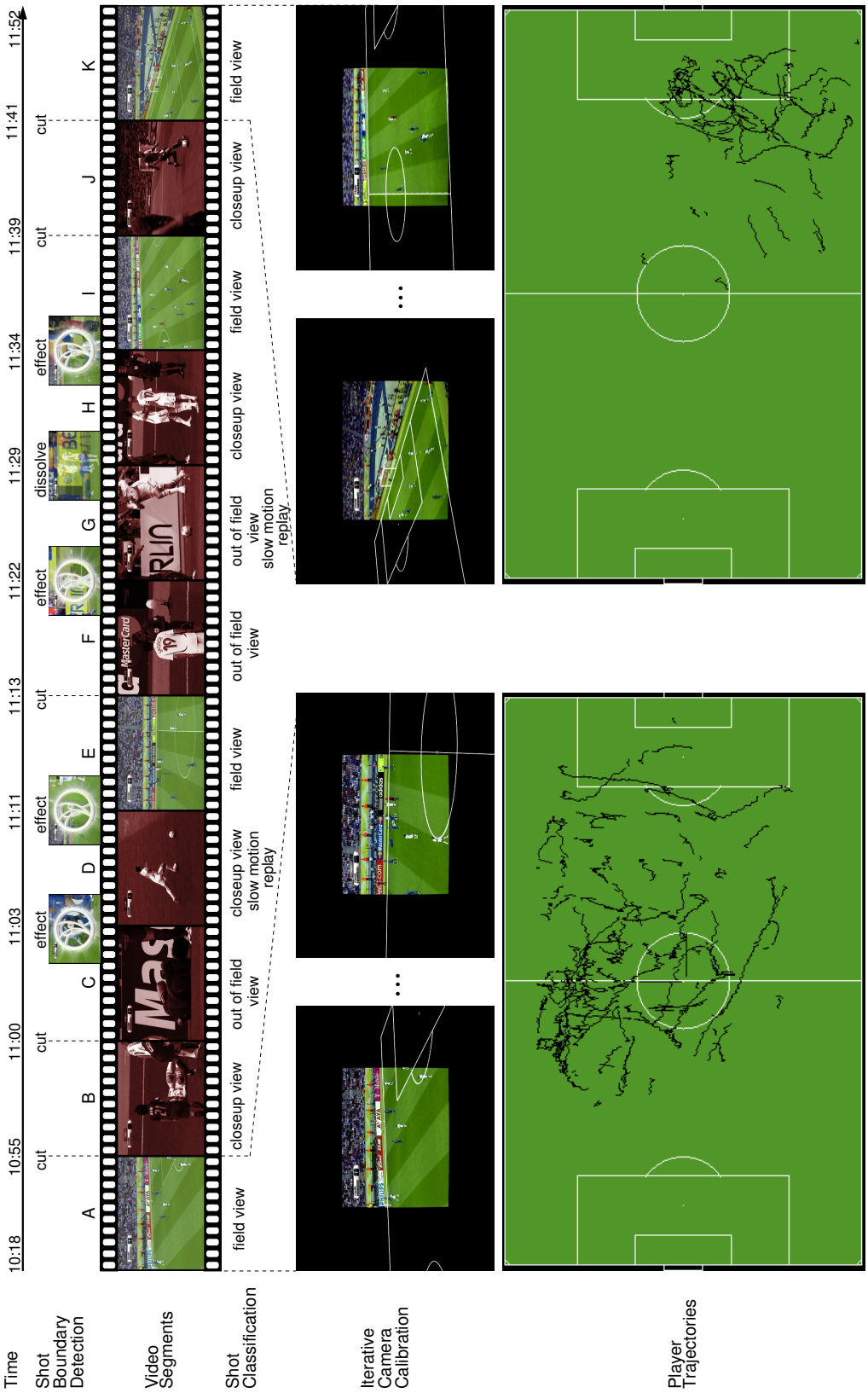


Figure 1.3 Example: The relevant video segments are determined and the camera parameters for these segments are estimated. From these parameters, the player positions and hence their trajectories within the video segments can be obtained.

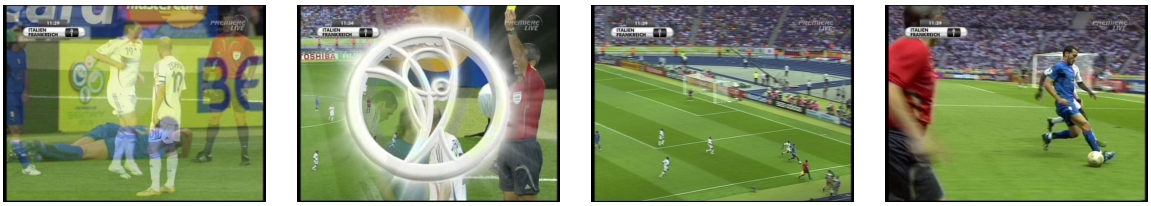
segments by partitioning the video stream at shot boundaries. In our example, the detected video segments are labeled with the capital letters A to K, and the shot boundary types are depicted above the film strip between two adjacent segments. Each segment is then classified to decide whether the shot was taken by the camera of interest (usually the main camera) and therefore may contain valuable information. The classification results are printed below the corresponding segments. For each relevant segment (A, E, I and K in our example) the camera parameters are estimated. This is visualized by the overlayed field model on the start and end frame of the segments A and K. In a further step, which is out of the scope of this thesis, the estimated camera parameters are then used to obtain the player trajectories on the playing field for each visible player within the considered video segment.

The example above illustrates the following requirements which are satisfied by the proposed camera tracking system.

- As in soccer, several types of shot boundaries are used, the system has to reliably detect all types of shot boundaries (see figure 1.4). This is fundamental, as missing a shot boundary possibly means to miss a relevant segment.

- Furthermore, the system has to robustly classify all the detected segments, which is essential if we are to decide whether a segment is relevant. The segment types used in soccer broadcastings are normal field views, out of field views and close up views. Slow motion replays are a special case, as they may contain information about the camera of interest. However, this information is out of the time line and therefore without value for the tracking task. Only field views are considered for further processing.

- Finally, the system has to initialize the tracking task, which robustly estimates the camera parameters for the whole segment until a shot boundary is detected. Since the cameras are not accessible or controlable and therefore neither partial camera calibration nor active localization is possible, the design of this step is important for the robustness and accuracy of the whole tracking system.



(a) (b) (c) (d)

Figure 1.4 Shot Boundary Types: (a) Dissolve, (b) Effect and (c,d) Cut.

Besides these difficulties, there are a number of additional issues concerning broadcasted video streams and the characteristics of soccer in general that have not been discussed in the example above. These can mostly be attributed to the fact that broadcasted videos are produced for entertainment purposes, and neither the hardware infrastructure nor the camera configuration or camera work are optimized for computer vision applications.

- The illumination conditions produce spatial and temporal inhomogeneities due to the large field size ($\sim 105m \times 68m$) and the fact that soccer is played outside. Day and night-time recordings result in different appearances of the field and the players due to shadows or sunlight reflections (see figure 1.5(a)). This leads to high variances and mixtures in object color distributions.

- According to FIFA regulations [FIF07], the field size in soccer may vary from stadium to stadium, which leads to two additional unknown parameters to be estimated.

- Field lines can be occluded by players, which complicates camera localization due to missing or even wrong features (see figure 1.5(b)).

- Due to the fast nature of soccer, rapid camera motion and zooming leads to blurry images and diffuse edges (see figure 1.5(c)).

- The television broadcasting norms for analog broadcasting, namely *PAL*, *SECAM* and *NTSC*, as well as digital broadcasting, *DVB-T*, *DVB-C* and *DVB-S*, are all using slightly modified versions of the YUV colormodel. As the human eye is more sensitive to changes in luminance (Y) than to changes in color or chrominance (UV), color information is reduced considerably to compress the data. Together with the effects caused by inhomogeneous illumination, the use of standard color segmentation becomes unreliable.

- Television broadcasting norms use interlaced frames with low resolutions (PAL: 720×576 and NTSC: 720×480). To reduce imprecise edges, only one of the two fields (even or odd lines) can be used per frame, which halves the image height and further reduces the image resolution. High definition videos are becoming more popular; older video tapings, however, are only available at these low resolutions.

A practically applicable system must be able to overcome these issues. Furthermore, the system must be able to run reliably with minimal operator interaction. It must be able to detect failures and reinitialize itself. Above all, it must be able to run near real-time, as a full length

soccer game consists of at least 90 minutes of video footage that has to be processed in a reasonable amount of time.

(a) (b) (c)

Figure 1.5 Difficult scenes: Inhomogene illumination due to (a) hard shadows, (b) almost no visible line features and (c) blurry images due to fast camera motion.

1.3 Our Approach

The main task to be solved is the camera tracking, i.e. the estimation of the intrinsic and extrinsic parameters of the camera model that is needed to describe the mapping between the world and the physical camera. Estimating all parameters simultaneously for all frames is infeasible, inefficient and unnecessary, as most camera parameters stay constant throughout the game. The positions of cameras in soccer broadcasts are fixed during the game as they are attached to tripods around the field. Their orientation and zoom factor (focal length) on the other hand are constantly changing to track the game. Considering this configuration and making some reasonable assumptions, the parameters are divided into two groups. The unknown *constant parameters*, that are determined beforehand in a mostly automated process, and the varying (dynamic) parameters, that have to be estimated in each frame during the tracking process.

Before being able to start the camera tracking, we first have to identify scenes in the video stream that originate from the camera of interest. We will now provide an overview of the process of the camera parameter estimation. The task has been divided into two stages that involve the interaction of several modules as depicted in figure 1.6.

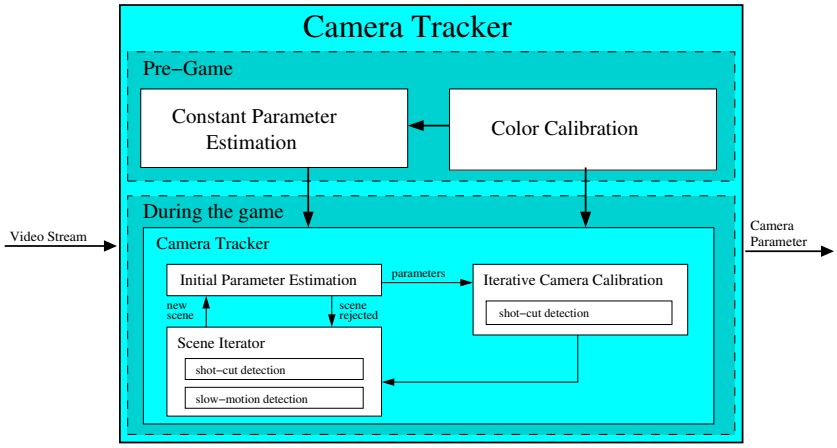


Figure 1.6 The *Camera Tracker* module

- In the *Pre-game* stage, the *Constant Parameter Estimation* determines all necessary parameters and the field color using multiple views from the respective camera. The user has to select several images from the camera of interest and provide point correspondences for each image. This is the only necessary interaction between the user and the system. The constant parameters, including their uncertainties, are then estimated automatically.

- During the game, the *Camera Tracker* iterates through all sequences that correspond to the camera of interest, and starts the *Iterative Camera Calibration* process.

  To do so, the *Scene Iterator* module segments the video-stream at shot boundaries into small sequences, the *video segments*. It iterates successively through all video segments and skips *out of field views*, *close up views* and *slow motion replays*. The remaining segments are potentially from the camera of interest and are therefore passed on to the *Initial Parameter Estimator*.

  The *Initial Parameter Estimator* searches for the initial dynamic parameters (pan, tilt and zoom) of a segment by using extracted image lines. This is repeated until consistent parameters are found. If no parameters are found, i.e. the image lines are inconsistent with the expected projections of the model lines, the video segment is either not from the camera of interest or does not contain scenes of the penalty or goal area. In this case, the segment is rejected and the *Scene Iterator* searches for the next potential video segment from the camera of interest. If initial parameters could be found, the *Iterative Camera Calibration* is initiated, which successively estimates the camera parameters for the sequence until a shot boundary is detected.

Camera Parameter Estimation

Since the image formation process maps 3D world coordinates to 2D image coordinates, information is lost and the inverse transformation can not be uniquely determined from a single image without a priori knowledge. In standard calibration methods, this a priori knowledge is usually provided by a 3D model of an object mapped in the image. However, in TV broadcasts of soccer games, the cameras are neither accessible nor controllable. Therefore, no calibration device can be used to obtain the camera parameters by identifying the corresponding 2D projections of known real world features in the image. Thus, standard calibration methods are infeasible.

In the case of soccer, the field provides a set of field markings specified by the official soccer rules [FIF07] that can serve as good features to carry out camera calibration. The camera calibration is done using a *field model* that describes the appearance of the field with respect to the camera of interest. Therefore, the field can be seen as a large calibration device. As mentioned earlier, the field size in soccer is not completely specified by the official rules, and therefore has to be provided by the user or estimated beforehand in the *Constant Parameter Estimation* module.

As depicted in figure 1.7, the basic idea of camera parameter estimation is to find the parameters (right) leading to the best match (center) between the environment model (left bottom) and the image (left top).

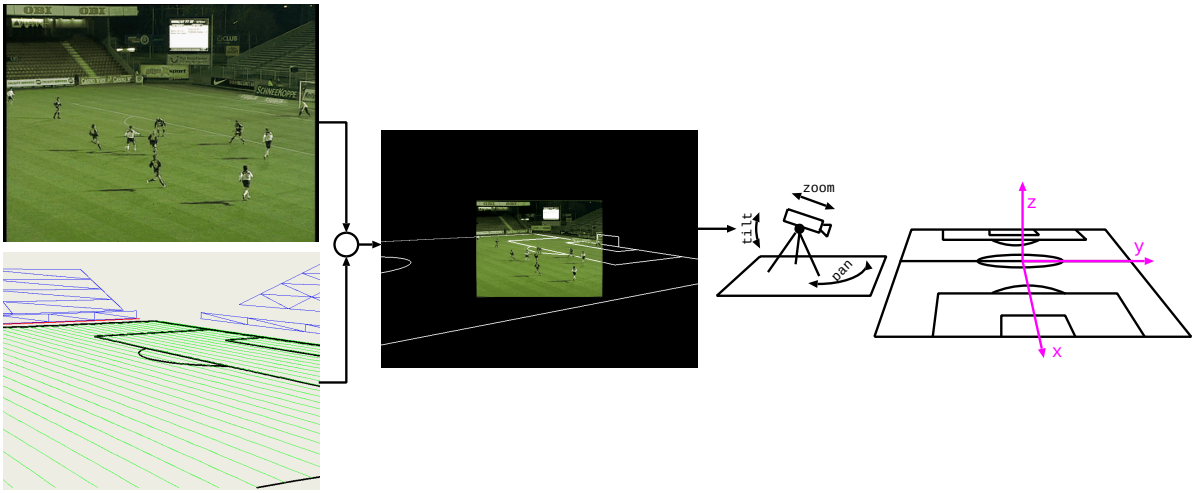


Figure 1.7 Model-based camera parameter estimation

Altough, this seems to be a straight foreward approach, the main challenge is to design an estimator that can robustly and accurately track the camera parameters, despite the challenges described in section 1.2 and the possibility of temporarily absent line features in the image.

1.4 Main Contributions

The achievements of this thesis are twofold and contribute to the camera calibration and tracking of sports games from TV broadcasts with a special focus on soccer.

First, this thesis presents a fast and robust camera tracking method that is capable of tracking the camera over long periods of time for uncut video streams. The tracking task is integrated into the Kalman Filter framework, and uses the following methods to cope with noisy and erroneous measurements, that are common in natural and highly dynamic environments:

- Parameters are predicted using the optical flow information between two subsequent images by exploiting domain knowledge about the camera configuration. This allows the tracking task to bridge gaps, where no field lines are visible, by providing an accurate prediction for short periods of time.

- The parameter update which uses detected field line features in the image, is performed by the so-called robust M-Estimators that are integrated into the commonly used Gauss-Newton estimator. Furthermore, we propose a new weight function that leads to an accurate final estimate by changing the shape of the weight function in addition to the clipping threshold in every iteration.

Second, we present a vision framework that is capable of handling raw video streams of soccer games from TV broadcasts. The system automatically segments and classifies the video stream into scenes corresponding to the camera of interest and scenes corresponding to other cameras or slow motion replays. The camera parameters for relevant segments are initialized and tracked as long as they last.

Combining the automated scene segmentation with the automated camera initialization and the robust camera tracking method, we provide a fast, reliable and accurate system that can be used by subsequent processes to obtain real world measurements from image observations, e.g. for estimating the player and ball trajectories on the field plane. The only requirements are cameras with fixed positions and a structured environment that can be described by a 3D model.

The camera tracking system is an essential part of the ASPOGAMO system (Automated SPOrts Game Analysis MOdel), which is capable of extracting player trajectories from soccer broadcasts [BBG+06, BGB+07, GBvHH+07, BBG+08]. ASPOGAMO has been presented to the public during RoboCup 2006 in Bremen, showing the practical applicability of the camera tracking system on live broadcasts of games from the FIFA World Cup 2006.

1.5 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 introduces some fundamental preliminaries and the used notation. The discussion includes point representations in 3D, homogeneous coordinates, elementary transformations, the Gaussian Normal distribution and its linear and non-linear propagation.

Chapter 3 introduces the pinhole camera model used, which is a fundamental block of this thesis. The whole transformation pipeline from image to world coordinates and vice versa is explained in detail, and the pinhole camera model is presented as a linear transformation in projective space.

Chapter 4 introduces important state estimation methods for linear and nonlinear systems. The differences and similarities between several estimation methods are presented. Furthermore, several common robust estimators are explained. This chapter provides the foundation for the recursive camera calibration process proposed in the next chapter.

Chapter 5 is the main part of this thesis and presents the camera tracking framework, that is capable to process raw videos of soccer games from TV-broadcasts. All modules and their interaction are explained in detail.

Chapter 6 summarizes the main contributions of this thesis and points out possible future directions.

Chapter 2 Preliminaries

This chapter introduces some fundamental preliminaries. Particular point representation in 3D, elementary Transformations, Gaussian normal distribution and its linear and nonlinear propagation.

2.1 Points and Transformations in 3D

Points in Euclidean 3-space are represented by a three dimensional column vector $\breve{\boldsymbol{p}} = (x, y, z)^T$. Several transformations can be represented as 3×3 matrices, so that the transformation itself becomes a matrix-vector multiplication.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

With this representation for example, it is possible to write rotations, scaling, shearing and arbitrary combinations of these entities as matrices. Successive transformations can easily be composed into a single matrix by simple matrix-matrix multiplication. For example:

$$\begin{aligned} \breve{\boldsymbol{p}}' &= A_1 \cdot \breve{\boldsymbol{p}} \\ \breve{\boldsymbol{p}}'' &= A_2 \cdot \breve{\boldsymbol{p}}' \end{aligned}$$

$\breve{\boldsymbol{p}}''$ can be written as:

$$\begin{aligned} \breve{\boldsymbol{p}}'' &= A_2 \cdot (A_1 \cdot \breve{\boldsymbol{p}}) = \underbrace{(A_2 \cdot A_1)}_{=A} \cdot \breve{\boldsymbol{p}} \\ &= A \cdot \breve{\boldsymbol{p}} \end{aligned}$$

However there exist a major disadvantage of this representation. Neither translations nor projections in 3D can be written as 3×3 matrices. This disadvantage can be eliminated by embedding the Euclidean 3-space $\mathbb{R}^3$ into the corresponding *Projective Space* $\mathbb{P}^3$ by introducing a fourth coordinate; the *homogeneous coordinate*. A brief description of this concept is given in the next section.

2.2 Homogeneous Coordinates

A homogeneous point in $\mathbb{P}^3$ is represented by a four dimensional column vector $(x, y, z, w)^T$. The homogeneous point $\boldsymbol{p} = (x, y, z, w)^T$ represents the Euclidean point $\breve{\boldsymbol{p}} = \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)^T$.

$$\lambda \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \equiv \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{pmatrix}, \lambda \neq 0 \tag{2.1}$$

The scale factor λ indicates that homogeneous points are invariant to non-zero scale. The rule for converting homogeneous points into Euclidean points and vice versa can directly be derived from equation (2.1), aside from homogeneous points with $w = 0$. Those points have no Euclidean counterpart and are called *ideal points*. They represent points lying at infinity (or lying on the *plane at infinity*). Since ideal points are invariant to translations, they can be seen intuitively as vectors.

2.2.1 Transformations in 3D

As mentioned above rigid transformations can be achieved now by multiplying points with 4×4 matrices. This section shows some fundamental transformations.

Translation

The translation of a point by a vector $\breve{\boldsymbol{t}} = (t_x, t_y, t_z)^T$ is given by the following matrix:

$$T\left(\breve{\boldsymbol{t}}\right) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

Rotation

The basis rotations around the three axis of the coordinate frame are given by:

$$R_x\left(\alpha\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\alpha\right) & -\sin\left(\alpha\right) & 0 \\ 0 & \sin\left(\alpha\right) & \cos\left(\alpha\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

$$R_y\left(\beta\right) = \begin{pmatrix} \cos\left(\beta\right) & 0 & \sin\left(\beta\right) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\left(\beta\right) & 0 & \cos\left(\beta\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

$$R_z\left(\gamma\right) = \begin{pmatrix} \cos\left(\gamma\right) & -\sin\left(\gamma\right) & 0 & 0 \\ \sin\left(\gamma\right) & \cos\left(\gamma\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.5}$$

A general rotation (rotating around an arbitrary axis by an arbitrary angle) can be achieved by joining these three basis rotations to a general rotation. The three rotation angles α, β and γ, which are the parameters of the general rotation, depend on the order of the basis rotations. In this thesis a general rotation is achieved as following: After a rotation around the z-axis with an angle of γ followed by a rotation around the y-axis with an angle of β a rotation around the x-axis with an angle of α is performed. The compiled rotation matrix is given as follows:

$$
\begin{aligned}
R(\alpha, \beta, \gamma) &= R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma) \\
&= \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
r_{11} &= \cos(\beta)\cos(\gamma) \\
r_{12} &= -\cos(\beta)\sin(\gamma) \\
r_{13} &= \sin(\beta) \\
r_{21} &= \sin(\alpha)\sin(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma) \\
r_{22} &= -\sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) \\
r_{23} &= -\sin(\alpha)\cos(\beta) \\
r_{31} &= -\cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) \\
r_{32} &= \cos(\alpha)\sin(\beta)\sin(\gamma) + \sin(\alpha)\cos(\gamma) \\
r_{33} &= \cos(\alpha)\cos(\beta)
\end{aligned} \tag{2.6}
$$

The rotation angles (α, β, γ) can be derived from R by following equations:

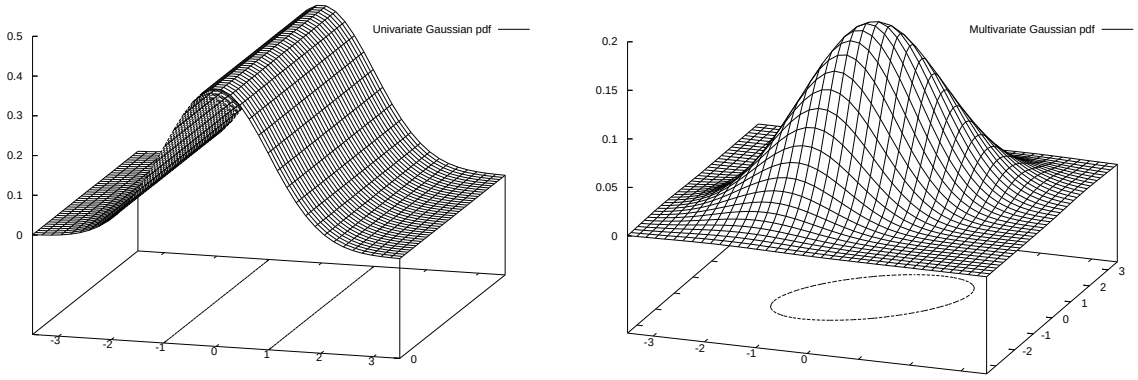
$$
\begin{aligned}
\cos(\beta) &= \sqrt{r_{11}^2 + r_{12}^2} \\
\alpha &= \operatorname{atan2}\left(-\frac{r_{23}}{\cos(\beta)}, \frac{r_{33}}{\cos(\beta)}\right) \\
\beta &= \operatorname{atan2}(r_{13}, \cos(\beta)) \\
\gamma &= \operatorname{atan2}\left(-\frac{r_{12}}{\cos(\beta)}, \frac{r_{11}}{\cos(\beta)}\right)
\end{aligned} \tag{2.7}
$$

Where $\mathrm{atan2}\,(y, x)$ is similar to $\arctan\left(\frac{y}{x}\right)$, but considers the signs of x and y to determine the correct quadrant. The problem cannot be solved uniquely ([Sla99]), for the case $\cos(\beta) \neq 0$ two solutions exists. For the special case of $\cos(\beta) = 0$ an infinite number of solutions exist. In this case the same rotation can be achieved with different combinations of α and γ.

2.3 Gaussian Normal Distribution

As sensor data are usually afflicted with measurement errors, these errors must be taken into account by the estimation process to achieve high accuracy and robustness. To do so the error distribution is described by a stochastic model.

As measurement errors are a summation of several error sources, which itself are randomly distributed according to an unique but unknown stochastic distribution, and according to the *central limit theorem* it is reasonable to assume an overall Gaussian normal distribution.



(a) (b)

Figure 2.1 Gaussian normal distribution: (a) univariate and (b) bivariate

The univariate Gaussian normal distribution is uniquely specified by its mean $\bar{x} = E[x]$ and variance $\sigma_x^2 = E\left[(x - \bar{x})^2\right]$, where $E[\cdot]$ represents the *expected value*. Its probability density function (*pdf*) $p(x)$ is defined by:

$$p(x) = N\left(x; \bar{x}, \sigma_x^2\right) = \frac{1}{\sqrt{2\pi}\sigma_x} e^{-\frac{1}{2}\left(\frac{x-\bar{x}}{\sigma_x}\right)^2} \tag{2.8}$$

In most cases measurement and model parameters are multidimensional and therefore are described by multivariate Gaussian normal distributions. In the multivariate case $\boldsymbol{x}$ and $\bar{\boldsymbol{x}}$ are n-dimensional vectors and Σ_{xx} is a symmetric positive semidefinite $n \times n$ matrix. The pdf for

a multivariate Gaussian normal distribution is defined by:

$$p(\boldsymbol{x}) = N(\boldsymbol{x}; \bar{\boldsymbol{x}}, \Sigma_{xx}) = \frac{1}{\sqrt{|2\pi\Sigma_{xx}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\bar{\boldsymbol{x}})^T \Sigma_{xx}^{-1} (\boldsymbol{x}-\bar{\boldsymbol{x}})} \tag{2.9}$$

The notation $\boldsymbol{x} \sim N(\bar{\boldsymbol{x}}, \Sigma_{xx})$ describes an n-dimensional Gaussian normal distributed random variable $\boldsymbol{x}$ with mean $\bar{\boldsymbol{x}} = E[\boldsymbol{x}]$ and covariance matrix $\Sigma_{xx} = E\left[(\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^T\right]$. The covariance matrix Σ_{xx} is a $n \times n$ symmetric and positive semidefinite matrix with following structure:

$$\Sigma_{xx} = \begin{pmatrix} Cov(x_1, x_1) & Cov(x_1, x_2) & \dots & Cov(x_1, x_n) \\ Cov(x_2, x_1) & Cov(x_2, x_2) & \dots & Cov(x_2, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(x_n, x_1) & Cov(x_n, x_2) & \dots & Cov(x_n, x_n) \end{pmatrix}$$

$$\begin{aligned} Cov(x_i, x_i) &= \sigma_i^2 = E\left[(x_i - \bar{x}_i)^2\right] \\ Cov(x_i, x_j) &= Cov(j, i) = E\left[(x_i - \bar{x}_i)(x_j - \bar{x}_j)\right] \end{aligned} \tag{2.10}$$

where $Cov(x_i, x_j)$ is the covariance of component x_i with component x_j.
Note, that the covariance matrix can be written as:

$$\begin{aligned} \Sigma_{xx} &= E\left[(\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^T\right] \\ &= E\left[\boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{x}\bar{\boldsymbol{x}}^T - \bar{\boldsymbol{x}}\boldsymbol{x}^T + \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T\right] \\ &= E\left[\boldsymbol{x}\boldsymbol{x}^T\right] - E[\boldsymbol{x}]\bar{\boldsymbol{x}}^T - \bar{\boldsymbol{x}}^T E[\boldsymbol{x}] + \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T \\ &= E\left[\boldsymbol{x}\boldsymbol{x}^T\right] - \bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^T \end{aligned} \tag{2.11}$$

For the univariate case it follows:

$$\sigma_x^2 = E\left[x^2\right] - E[x]^2 \tag{2.12}$$

This representation, for example, is important for the efficient determination of the variance images (see section 5.1.2.1).

2.3.1 Propagation of Gaussian pdfs

Parameter estimation algorithms are usually fitting a model to a set of measurements, by minimizing a cost function. As the measurements are usually in a different coordinate system than the parameters, the fitting process has to transform the measurements from the *observation space* into the *parameter space* or vice versa, to take their uncertainties into account. As both, the parameters and measurements are usually described by Gaussian pdfs, it is necessary to transform the whole pdf to a new pdf in the other coordinate frame.

Formally, this problem can be described as follows: Given a n-dimensional random variable $\boldsymbol{x} \sim N(\bar{\boldsymbol{x}}, \Sigma_{xx})$ and a function $h : \mathbb{R}^n \to \mathbb{R}^m$, the parameters $\bar{\boldsymbol{z}}$ and Σ_{zz} of the m-dimensional random variable $\boldsymbol{z} \sim N(\bar{\boldsymbol{z}}, \Sigma_{zz})$ with $\boldsymbol{z} = h(\boldsymbol{x})$ are to be determined.

2.3.1.1 Linear Transformation

If the transformation function h is linear, it can be written as: $h(\boldsymbol{x}) = H\boldsymbol{x}+\boldsymbol{b}$, with $H \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} \in \mathbb{R}^m$. The parameters $\bar{\boldsymbol{z}}$ and Σ_{zz} of the transformed pdf can be derived as follows:

$$\begin{aligned}
\bar{\boldsymbol{z}} &= E[\boldsymbol{z}] \\
&= E[h(\boldsymbol{x})] \\
&= E[H\boldsymbol{x} + \boldsymbol{b}] \\
&= HE[\boldsymbol{x}] + \boldsymbol{b} \\
&= H\bar{\boldsymbol{x}} + \boldsymbol{b} \qquad (2.13)
\end{aligned}$$

$$\begin{aligned}
\Sigma_{zz} &= E\left[(\boldsymbol{z} - \bar{\boldsymbol{z}})(\boldsymbol{z} - \bar{\boldsymbol{z}})^T\right] \\
&= E\left[(H\boldsymbol{x} + \boldsymbol{b} - (H\bar{\boldsymbol{x}} + \boldsymbol{b}))(H\boldsymbol{x} + \boldsymbol{b} - (H\bar{\boldsymbol{x}} + \boldsymbol{b}))^T\right] \\
&= E\left[(H\boldsymbol{x} - H\bar{\boldsymbol{x}})(H\boldsymbol{x} - H\bar{\boldsymbol{x}})^T\right] \\
&= E\left[(H(\boldsymbol{x} - \bar{\boldsymbol{x}}))(H(\boldsymbol{x} - \bar{\boldsymbol{x}}))^T\right] \\
&= HE\left[(\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^T\right] H^T \\
&= H\Sigma_{xx}H^T \qquad (2.14)
\end{aligned}$$

2.3.1.2 Nonlinear Transformation

In practice transformation functions are rarely linear and therefore the transformed pdf is not Gaussian. In this case the transformed pdf has to be approximated by a Gaussian pdf. The most common methods for nonlinear propagation are shown below.

Linearization using first order Taylor series

Each derivable function can be approximated by a Taylor series about a given seed point. Neglecting all non-linear terms leads to the first order Taylor approximation, which can be geometrically interpreted as the tangent of the function at the seed point (see figure 2.2(a)).

$$h(\boldsymbol{x}) \approx h(\boldsymbol{x}_0) + J_h(\boldsymbol{x} - \boldsymbol{x}_0) \tag{2.15}$$

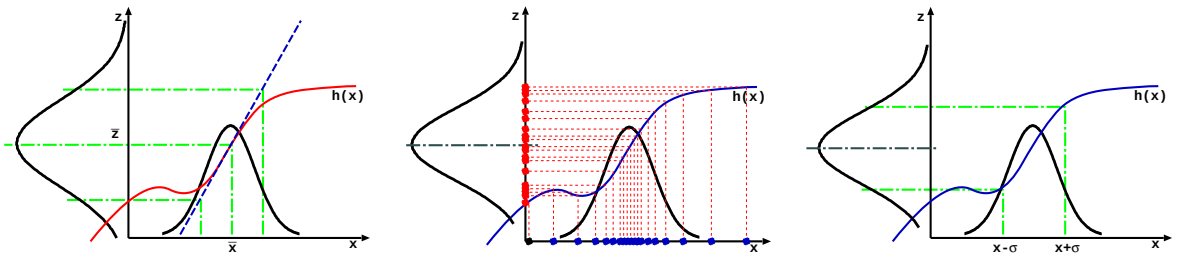
J_h is a $m \times n$ matrix containing the partial derivatives of h at the point $\boldsymbol{x}_0$ and is called the *Jacobian* or *Jacobi* matrix of h.

$$J_h = \frac{\partial h}{\partial \boldsymbol{x}}|\boldsymbol{x}_0 = \begin{pmatrix} \frac{\partial h_1}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial h_1}{\partial x_2}(\boldsymbol{x}_0) & \dots & \frac{\partial h_1}{\partial x_n}(\boldsymbol{x}_0) \\ \frac{\partial h_2}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial h_2}{\partial x_2}(\boldsymbol{x}_0) & \dots & \frac{\partial h_2}{\partial x_n}(\boldsymbol{x}_0) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial x_1}(\boldsymbol{x}_0) & \frac{\partial h_m}{\partial x_2}(\boldsymbol{x}_0) & \dots & \frac{\partial h_m}{\partial x_n}(\boldsymbol{x}_0) \end{pmatrix} \tag{2.16}$$

Converting equation (2.15) leads to following representation:

$$\begin{aligned} h(\boldsymbol{x}) &= h(\boldsymbol{x}_0) + J_h(\boldsymbol{x} - \boldsymbol{x}_0) \\ &= \underbrace{J_h}_{H}\boldsymbol{x} + \underbrace{h(\boldsymbol{x}_0) - J_h\boldsymbol{x}_0}_{\boldsymbol{b}} \\ &= H\boldsymbol{x} + \boldsymbol{b} \end{aligned} \tag{2.17}$$

and finally linear propagation (2.13) and (2.14) can be used.



(a) (b) (c)

Figure 2.2 Nonlinear propagation of Gaussian pdfs: (a) linearization method, (b) Monte Carlo method and (c) Unscented Transformation

As can be seen in figure 2.2(a), it is reasonable to select the mean $\bar{\boldsymbol{x}}$ of the original pdf as the

seedpoint, which leads to following parameters using equation (2.17):

$$\begin{aligned} \bar{\boldsymbol{z}} &= H\bar{\boldsymbol{x}} + \boldsymbol{b} \\ &= J_h\bar{\boldsymbol{x}} + h\left(\bar{\boldsymbol{x}}\right) - J_h\bar{\boldsymbol{x}} \\ &= h\left(\bar{\boldsymbol{x}}\right) \end{aligned} \tag{2.18}$$

$$\begin{aligned} \Sigma_{zz} &= H\Sigma_{xx}H^T \\ &= J_h\Sigma_{xx}J_h^T \end{aligned} \tag{2.19}$$

The drawback of this method is its inaccuracy since the transformed pdf is not approximated by a Gaussian, but the transformation function is approximated by a linear function and linear propagation is used. Transformation functions with high curvature at the seed point lead to considerably large errors making this method less capable.

Monte Carlo Method

The Monte Carlo method is a derivative free method which uses the true transformation function h for propagation. Unlike the linearization method, this method approximates the transformed non-Gaussian pdf by a Gaussian one.
First a big (> 1000) sample set is drawn from the original distribution $\boldsymbol{x}_i \sim N\left(\boldsymbol{x}; \bar{\boldsymbol{x}}, \Sigma_{xx}\right)$, then each point $\boldsymbol{x}_i$ is transformed using the transformation function into a new set $\boldsymbol{z}_i$ (see figure 2.2(b)).

$$\boldsymbol{z}_i = h(\boldsymbol{x}_i) \tag{2.20}$$

Depending on the number of samples, the transformed points $\boldsymbol{z}_i$ represent the transformed pdf more or less accurate. Finally the searched Gaussian is given by the sample mean $\bar{\boldsymbol{z}}$ and sample variance Σ_{zz}.

$$\begin{aligned} \bar{\boldsymbol{z}} &= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{z}_i \\ \Sigma_{zz} &= \frac{1}{n-1}\sum_{i=1}^{n}\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)\left(\boldsymbol{z}_i - \bar{\boldsymbol{z}}\right)^T \end{aligned} \tag{2.21}$$

Unfortunately, the transformation has to be evaluated for each sample, and therefore the major drawback of this method is its inefficiency.

Unscented Transformation

Another derivative free method which makes a good tradeoff between accuracy and efficiency is the *Unscented Transformation*. It was proposed by Julier and Uhlmann ([JU97]). The idea is similar to the Monte Carlo method, except the selection and the number of sample points. Only a small number of certain selected points, the *sigma points*, are used making this method more efficient than the Monte Carlo method and more accurate than the linearization method. First $2n$ sigma points are calculated from the original pdf, where n denotes the dimension of the pdf.

$$\begin{aligned} \boldsymbol{x}_i &= \boldsymbol{\sigma}_i + \bar{\boldsymbol{x}} \\ \boldsymbol{x}_{n+i} &= \boldsymbol{\sigma}_i - \bar{\boldsymbol{x}} \qquad\qquad 1 \leq i \leq n \end{aligned} \tag{2.22}$$

where $\boldsymbol{\sigma}_i$ is the i-th row or column of $\sqrt{n\Sigma_{xx}}$. The matrix square root can be computed using the *Cholesky-decomposition* or the eigenvalue-decomposition [PTVF92]. The latter delivers the scaled eigenvectors of the covariance matrix with the square roots of the corresponding eigenvalues.

Then the sigma points are transformed using the transformation function to a new sample set, and the transformed Gaussian is obtained by the sample mean and sample variance of this new sample set (see figure 2.2(c)).

The unscented transformation is superior compared to the linearization method for all continuous nonlinear transformations and does not need the derivatives of the transformation function.

In the context of this thesis the unscented transformation is used to propagate uncertainties through several nonlinear transformations. e.g. the position uncertainty from image points in pixel coordinate system to points on the field in world coordinate system.

2.3.2 Variance along a given direction

The variance along a direction, given by the vector $\boldsymbol{n}$ with unit length, referring to the multivariate Gaussian normal distribution $\boldsymbol{x} \sim N(\bar{\boldsymbol{x}}, \Sigma_{xx})$, is given by following equation:

$$\sigma_n^2 = \boldsymbol{n}^T \Sigma_{xx} \boldsymbol{n} \tag{2.23}$$

This can be seen after reinterpreting the problem as the problem of finding the variance of the distance d_x of points $\boldsymbol{x}$ drawn from the distribution $N(\bar{\boldsymbol{x}}, \Sigma_{xx})$, to the plane going through $\bar{\boldsymbol{x}}$

and perpendicular to $\boldsymbol{n}$. The distance is given by:

$$\begin{aligned} d_x &= \boldsymbol{n}^T (\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ &= \boldsymbol{n}^T \boldsymbol{x} - \boldsymbol{n}^T \bar{\boldsymbol{x}} \end{aligned} \tag{2.24}$$

Since the last term $\boldsymbol{n}^T \bar{\boldsymbol{x}}$ is constant, applying (2.14) leads to above equation (2.23).

Rewriting equation (2.23) by using the Eigenvalue decomposition [PTVF92] leads to the following equation:

$$\begin{aligned} \sigma_n^2 &= \boldsymbol{n}^T \Sigma_{xx} \boldsymbol{n} \\ &= \boldsymbol{n}^T U D \underbrace{U^T \boldsymbol{n}}_{\boldsymbol{n'}} \\ &= \boldsymbol{n'}^T D \boldsymbol{n'} \\ &= \sum_{i=1}^{n} n_i'^2 \cdot \lambda_i \end{aligned} \tag{2.25}$$

Where U is an $n \times n$ orthonormal matrix containing the Eigenvectors of Σ_{xx} in the columns, and D is a diagonal $n \times n$ matrix containing the corresponding Eigenvalues.

Since the matrix U is orthonormal, vector $\boldsymbol{n'}$ has also unit length. The variance σ_n^2 is a weighted average of the eigenvalues of Σ_{xx}, thus lying between the smallest and largest eigenvalue. It can be seen, that the eigenvalues are the variances along their corresponding eigenvectors, which are the principal axis of the distribution. This property is exploited in the so-called *Principle Component Analysis* or PCA (see [DHS00]).

2.3.3 Mahalanobis Distance

A commonly used normalized distance of a given point $\boldsymbol{y}$ relative to a normal distribution $N(\bar{\boldsymbol{x}}, \Sigma_{xx})$ is the *Mahalanobis Distance*. The squared Mahalanobis Distance is defined as follows:

$$d_M^2 (\boldsymbol{y}; \bar{\boldsymbol{x}}, \Sigma_{xx}) = (\boldsymbol{y} - \bar{\boldsymbol{x}})^T \Sigma_{xx}^{-1} (\boldsymbol{y} - \bar{\boldsymbol{x}}) \tag{2.26}$$

In the one-dimensional case it can be seen, that the squared Mahalanobis distance is the squared distance to the mean in units of the variance σ_x^2.

$$d_M^2 \left(y; \bar{x}, \sigma_x^2\right) = \left(\frac{y - \bar{x}}{\sigma_x}\right)^2 \tag{2.27}$$

In multidimensions, this interpretation can not be hold. Instead, it can be interpreted as the sum of the squared distance components in units of variances along the principal axis.

$$d_M^2\left(\boldsymbol{y}; \bar{\boldsymbol{x}}, \Sigma_{xx}\right) = \sum_{i=1}^{n} \frac{y_i'^2}{\lambda_i} \tag{2.28}$$

with $\boldsymbol{y'} = U\left(\boldsymbol{y} - \bar{\boldsymbol{x}}\right)$ and U containing the principal axis (eigenvectors) of Σ_{xx} as mentioned in section 2.3.2.

The set of all points with the same squared Mahalanobis distance c^2 satisfy

$$\left(\boldsymbol{y} - \bar{\boldsymbol{x}}\right)^T \Sigma_{xx}^{-1} \left(\boldsymbol{y} - \bar{\boldsymbol{x}}\right) = c^2 \tag{2.29}$$

which is the general equation of an ellipsoid centred at the point $\bar{\boldsymbol{x}}$. The direction and length of the semi axis of the ellipsoid are the eigenvectors and the square roots of the corresponding eigenvalues of the covariance matrix respectively.

For the univariate case the equation becomes a standard quadratic equation and has one or two solutions. In the multivariate case all points lie on the surface of a hyperellipsoid. Figure 2.1 (a) shows the univariate case with $d_M = 1$, where both equidistant points are drawn as two parallel lines [1]. Figure 2.1 (b) shows the bivariate (two-dimensional) case with $d_M = 1$, where the equidistant points form an ellipse which is displayed on the ground plane.

Mahalanobis Distance of two Gaussians

So far the Mahalanobis distance of an undisturbed point $\boldsymbol{y}$ to a Gaussian distribution was considered. But in practice $\boldsymbol{y}$ can also be disturbed by normally distributed noise. i.e. $\bar{\boldsymbol{y}} = \boldsymbol{y}$ and $\Sigma_{yy} \neq 0$. If both distributions are independent, the squared Mahalanobis distance between both is defined as follows:

$$d_M^2\left(\boldsymbol{x}, \boldsymbol{y}; \bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}, \Sigma_{xx}, \Sigma_{yy}\right) = \left(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}\right)^T \left(\Sigma_{xx} + \Sigma_{yy}\right)^{-1} \left(\bar{\boldsymbol{x}} - \bar{\boldsymbol{y}}\right) \tag{2.30}$$

It can be seen that the Mahalanobis distance of two Gaussians is the generalization of the Mahalanobis distance as defined previously. As undisturbed points $\boldsymbol{y}$ have a covariance matrix of $\Sigma_{yy} = 0$, the above equation reduces to (2.26).

[1] Both points are displayed as lines, because of the 3D plot of a 2D function.

Hypothesis Testing

Hypothesis testing provides a formal way to decide, to a certain level of confidence α, whether an assumption should be accepted or rejected. A common way to decide, whether a measurement corresponds to a given normal distribution or not, is the use its Mahalanobis distance to the distribution [BSF88].

The squared Mahalanobis distance d_M^2 of a n dimensional random variable $\boldsymbol{x} \sim N(\bar{\boldsymbol{x}}, \Sigma_{xx})$ is *chi-square* (χ^2) distributed with n degrees of freedom. This can be derived as follows: Suppose the square root of the Covariance matrix[2] Σ_{xx} is given by C with $\Sigma_{xx} = CC^T$.

Then, the Mahalanobis distance can be rewritten as:

$$\begin{aligned} d_M^2(\boldsymbol{x}; \bar{\boldsymbol{x}}, \Sigma_{xx}) &= (\boldsymbol{x} - \bar{\boldsymbol{x}})^T \left(CC^T\right)^{-1} (\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ &= (\boldsymbol{x} - \bar{\boldsymbol{x}})^T \left(C^{-T}C^{-1}\right) (\boldsymbol{x} - \bar{\boldsymbol{x}}) \\ &= \left((\boldsymbol{x} - \bar{\boldsymbol{x}})^T C^{-T}\right) \underbrace{\left(C^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}})\right)}_{=\boldsymbol{y}} \\ &= \boldsymbol{y}^T \boldsymbol{y} \end{aligned} \tag{2.31}$$

Using (2.13) and (2.14) leads to following mean and covariance matrix of $\boldsymbol{y}$:

$$\begin{aligned} \bar{\boldsymbol{y}} &= E\left[C^{-1}(\boldsymbol{x} - \bar{\boldsymbol{x}})\right] = C^{-1}E\left[(\boldsymbol{x} - \bar{\boldsymbol{x}})\right] = 0 \\ \Sigma_{yy} &= C^{-1}\Sigma_{xx}C^{-T} \\ &= C^{-1}\left(CC^T\right)C^{-T} \\ &= \left(C^{-1}C\right)\left(C^TC^{-T}\right) \\ &= I_n \end{aligned} \tag{2.32}$$

It can be seen, that the components y_i with $i \in [1; n]$ of $\boldsymbol{y}$ are now uncorrelated and have unit variance, i.e. $y_i \sim N(0, 1)$. $\boldsymbol{y}^T\boldsymbol{y}$ is the sum of n uncorrelated squared standard normal distributed random variables, thus $d_M^2(\boldsymbol{x}; \bar{\boldsymbol{x}}, \Sigma_{xx}) \sim \chi_n^2$.

The hypothesis H_0 that a point $\boldsymbol{x}$ belongs to a given normal distribution $N(\bar{\boldsymbol{x}}, \Sigma_{xx})$ can be tested with a given level of confidence α (probability of error) as follows:

$$d_M^2(\boldsymbol{x}, \bar{\boldsymbol{x}}, \Sigma_{xx}) < \chi_{n;\alpha}^2 \tag{2.33}$$

The hypothesis H_0 is accepted if the test criteria is true. The values for $\chi_{n;\alpha}^2$can be found in

[2] C can be obtained using the Cholesky factorization or the Eigenvalue decomposition [PTVF92].

χ^2 distribution tables. Commonly used values for α are $0.1, 0.05$ or 0.01. This test is used as the so-called validation gate in tracking [BSF88] or as termination criteria in MAP estimation (see chapter 4).

As all points with the same squared Mahalanobis distance describe an hyperellipsoid, the test can interpreted geometrically as following: All points satisfying the test are lying inside the ellipsoid described by equation (2.29) with $c^2 = \chi^2_{n;\alpha}$.

2.4 Conclusion

In this chapter we introduced transformations in Euclidean and projective space along with fundamental preliminaries for sensor processing algorithms. In particular, we introduced the multivariate Gaussian normal distribution, a commonly used distance measure based on it (i.e. the Mahalanobis distance) and the linear and several variants of the nonlinear transformation of normally distributed random variables.

In the next chapter, we will present the pinhole camera model with radial distortion, which is a fundamental part of this thesis.

Chapter 3 Camera Model

A camera maps continuous points from 3D world coordinates to discrete 2D points on the image: the pixels. As this mapping produces a two-dimensional image representing a three-dimensional scene, information gets lost and the mapping is called *projection*. Understanding this mapping is essential for vision based localization and state estimation.

A *camera model* describes this mapping mathematically more or less accurate, depending on the complexity of the used camera system and the underlying camera model. Depending on the application, a camera model has to compromise with the opposing goals of accuracy and efficiency.

As multiple camera types and systems are available, each type/system is described best through an appropriate camera model. For example, cameras with telecentric objectives are best described in most cases through *parallel* projection. Whereas for omnivision camera systems, which are using convex mirrors for grabbing 360° panoramic views, more complex camera models dealing with joint distortions caused by the objectives as well as by the mirrors are used. In this thesis, where video streams from standard TV broadcasting cameras are used, the best choice is a pinhole camera model with radial distortion, which is described in the next section.

3.1 Pinhole Camera Model with Radial Distortion

The pinhole camera model is the mostly used camera model for standard camera systems, since it makes a good tradeoff between accuracy and efficiency. It is based on the *central projection*, which assumes, that all light-rays building the image go through one single point, called the *central point*, *camera center*, *focus*, *focal point*, *combustion point* or *pinhole*[1], and are mapped to single points on the image. Both assumptions can not hold for most real camera systems making this model to a more or less accurate approximation of the real mapping. The

[1] All these terms are interchangeable and will be used alternated throughout this thesis

effects caused by construction conditions and limited production accuracies are lens distortions, color abberations, wrong focus, etc. Except the lens distortion, which can cause considerable pixel displacements on the image, most of the remaining effects are very small and negligible for most applications. The first camera model dealing with lens distortion is the *pinhole camera model with radial distortion* and was described by Lenz [LT88].

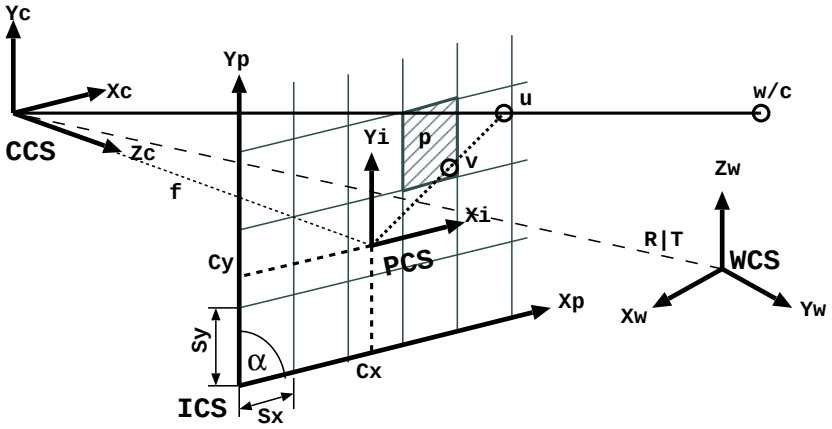


Figure 3.1 Pinhole camera model with radial distortion

This model can be described completely by 13 parameters which are divided into two groups. The first group, containing the external or *extrinsic* parameters, describes the rigid transformation of the camera coordinate system (*CCS*) with respect to the world coordinate system (*WCS*). Since a rigid transformation is a combination of a general rotation R and a translation T, as described in section 2.2.1, it has six parameters, three for the general rotation (α, β, γ) and three for the translation (t_x, t_y, t_z). The second group contains the *intrinsic* parameters, which describe the perspective projection of points from the camera coordinate system (*CCS*) into points in the pixel coordinate system (*PCS*) and are listed below

$\mathbf{f}$**:** The *Focal Length* is the distance of the *image plane* to the center of projection.

$(\mathbf{C_x}, \mathbf{C_y})^{\mathbf{T}}$**:** The *Principal Point* is the intersection point of the ray perpendicular to the image plane and passing through the camera center (the *principal ray* or *principal axis*).

$\mathbf{S_x}, \mathbf{S_y}$**:** Metric size of a sensor element on the image plane.

κ**:** The radial distortion coefficient describes the mapping between the ideal projected point $\boldsymbol{u}$ and the displaced true point $\boldsymbol{v}$ in image coordinate system (ICS).

$\mathbf{s}$**:** The *skew* parameter describes the rectangularity of the sensor elements (pixels) and is a function of the angle between the sensor axes.

The transformation pipeline describing the mapping of three dimensional points in WCS into two dimensional discrete points in PCS is decomposed into 4 steps and described in following subsections.

3.1.1 Rigid Transformation in Camera Coordinate Frame

As only the relative position of a point to the camera is important for its projection, a point $\breve{\boldsymbol{w}} = (w_x, w_y, w_z)^T$ in WCS has to be transformed first to a point $\breve{\boldsymbol{c}} = (c_x, c_y, c_z)^T$ in CCS by the rigid transformation described by the extrinsic parameters of the camera.

$$\breve{R}_c = \breve{R}(\alpha, \beta, \gamma) = \begin{pmatrix} r_1 1 & r_1 2 & r_1 3 \\ r_2 1 & r_2 2 & r_2 3 \\ r_3 1 & r_3 2 & r_3 3 \end{pmatrix}, \breve{\boldsymbol{t}}_c = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$$
$$\breve{\boldsymbol{c}} = \breve{R}_c(\breve{\boldsymbol{w}} - \breve{\boldsymbol{t}}_c) \tag{3.1}$$

where $\breve{\boldsymbol{t}}_c$ and $\breve{R}_c$ represent the position and orientation of the CCS in respect to the WCS. The above equation can be written more elegantly in homogeneous Coordinates as follows:

$$\boldsymbol{c} = \underbrace{R \cdot T}_{=Rt} \cdot \boldsymbol{w}$$

with

$$Rt = \begin{pmatrix} \breve{R}_c & -\breve{R}_c \cdot \breve{\boldsymbol{t}}_c \\ \mathbf{0}_3^T & 1 \end{pmatrix} \tag{3.2}$$

3.1.2 Perspective Projection

The perspective projection of a point $\breve{\boldsymbol{c}}$ in CCS to the corresponding point $\breve{\boldsymbol{u}}$ in Image Coordinate System (*ICS*) is performed as follows:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \frac{c_x f}{c_z} \\ \frac{c_y f}{c_z} \end{pmatrix} \tag{3.3}$$

The corresponding transformation in homogeneous coordinates, mapping homogeneous points

in $\mathbb{P}^3$ to homogeneous points in $\mathbb{P}^2$, is given by:

$$\boldsymbol{u} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \boldsymbol{c} \tag{3.4}$$

3.1.2.1 Radial distortion

The distortion caused by lens aberrations is described by a radial distortion model with one coefficient. Albeit camera models exists, describing the lens distortion more complex [WCH92] (e.g. several radial and tangential distortion coefficient), one coefficient is sufficiently accurate in the context of this thesis, as it describes about 90% of the total lens distortion ([Zha00, Tsa87, MT96]).

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \frac{2u_x}{1+\sqrt{1-4\kappa(u_x^2+u_y^2)}} \\ \frac{2u_y}{1+\sqrt{1-4\kappa(u_x^2+u_y^2)}} \end{pmatrix} \tag{3.5}$$

Negative values for κ lead to pincushion distortions, whereas positive values lead to barrel distortions (see figure 3.2).

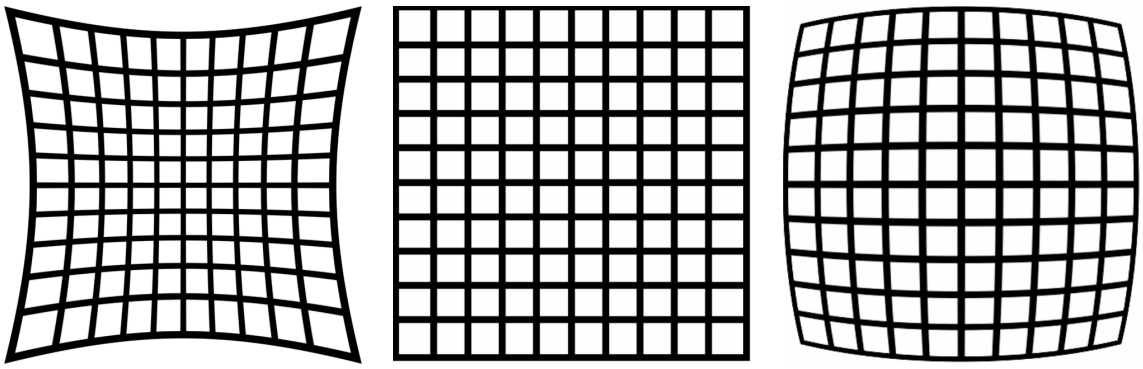


Figure 3.2 Radial distortion effects: pincushion $\kappa > 0$ (left), undistorted $\kappa = 0$ (center) and barrel distortion $\kappa < 0$ (right)

The above description of the radial distortion allows a simple inverse mapping.

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \frac{v_x}{1+\kappa(v_x^2+v_y^2)} \\ \frac{v_y}{1+\kappa(v_x^2+v_y^2)} \end{pmatrix} \tag{3.6}$$

3.1.2.2 Transformation from Image Coordinates to Pixel Coordinates

Finally the image point has to be transformed into the pixel coordinate system (*PCS*).

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} \frac{v_x}{S_x} + \frac{sv_y}{S_x} + C_x \\ \frac{v_y}{S_y} + C_y \end{pmatrix} \tag{3.7}$$

where the parameter s describes the skew of the pixel and depends on the tangent of the angle between the x and y axes of the sensor.

$$s = -\frac{1}{\tan(\alpha)} = -\cot(\alpha) \tag{3.8}$$

As sensors are manufactured with high precision, it is usually assumed that the pixels are rectangular and s is set to zero.
The corresponding homogeneous representation of above transformation is given by

$$\boldsymbol{p} = \begin{pmatrix} \frac{1}{S_x} & \frac{s}{S_x} & C_x \\ 0 & \frac{1}{S_y} & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \boldsymbol{v} \tag{3.9}$$

3.1.3 Perspective Projection in Homogeneous Coordinates

Apart from the radial distortion, the whole projection pipeline can be described by a linear transformation in *projective space*.

$$\boldsymbol{p} = P\boldsymbol{w} \tag{3.10}$$

The 3×4 matrix P mapping points $\boldsymbol{w}$ from $\mathbb{P}^3$ (homogeneous points in WCS) to point $\boldsymbol{p}$ in $\mathbb{P}^2$ (homogeneous points in PCS) is called the *camera projection matrix* and is a composition of equations (3.2), (3.4) and (3.9).

This matrix is usually decomposed into a 3×3 upper triangular matrix K, called the *camera matrix*, and a 3×4 matrix describing the rigid transformation of the CCS in respect to the WCS.

$$\begin{aligned} P &= K \cdot \tilde{R}t \\ &= \begin{pmatrix} \frac{f}{S_x} & \frac{fs}{S_x} & C_x \\ 0 & \frac{f}{S_y} & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \breve{R}_c & -\breve{R}_c \cdot \breve{\boldsymbol{t}}_c \end{pmatrix} \end{aligned} \tag{3.11}$$

It can be seen that the absolute values of f, S_x and S_y are not important and the ratios $f_x = \frac{f}{S_x}$ and $f_y = \frac{f}{S_y}$ are uniquely defining the camera matrix.
In practical applications the skew parameter is set to 0 and S_y is selected to an arbitrary non-zero value or obtained from the camera specifications. Thus the camera matrix can be completely specified by only 4 parameters.

$$K = \begin{pmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

Alternatively f_x can be replaced by $\theta \cdot f_y$, with $\theta := \frac{S_y}{S_x}$ being the aspect ratio of the sensor elements.

For cameras with varying focal length, it is usually assumed that all other intrinsic parameters stay constant and are known beforehand. Albeit this is not true, as changing the focal length causes a change in the radial distortion as well as a displacement of the principal point, the effects in the context of this thesis are small and can be neglected. For a camera with known pixel size, principal point and zero skew, the camera matrix can be written as the product of two matrices as follows:

$$\begin{aligned} K &= K_a \cdot K_p \\ &= \begin{pmatrix} \frac{1}{S_x} & 0 & C_x \\ 0 & \frac{1}{S_y} & C_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{3.13}$$

This representation is important, since image features can be transformed beforehand with the inverse of the affine transformation matrix K_a, and the further processing has to deal with the simplified camera matrix K_p. In this case we will refer K_p as the camera matrix and use the term K instead of K_p.

3.1.4 Camera Motion in Soccer Broadcasts

In soccer broadcasts, cameras are usually mounted on a tripod and therefore are fixed in their positions, while their orientation and zoom factor changes.

Assuming that the camera center does not change while rotating and zooming, and that the axes of the tripod (rotation) and the axes of the camera coordinate system coincides, and in the neutral position are parallel to the axes of the world coordinate system, the *pan* and *tilt* rotation results in the change of the γ and α angles of the camera respectively.
Figure 3.3 shows the supposed camera motion in soccer bradcasts, except the roll angle which is usually assumed to be constant.

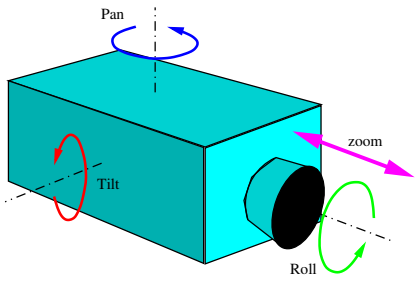


Figure 3.3 Camera motion in soccer broadcasts

However, figure 3.3 shows the pan, tilt and roll rotation in an intuitive way. As the camera coordinate frame coincides in the neutral position with the world coordinate frame, and the order of the basis rotations are selected as described in section 2.2.1, the real rotation is performed as shown in figure 3.4

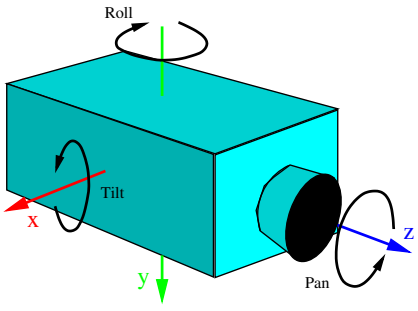


Figure 3.4 Camera rotation axes

Chapter 4 State Estimation

State estimation is a fundamental problem of sensor-based applications, as reliable analysis and decisions can only be made with the knowledge of the current system state.

The system state is described by the n-dimensional *state vector* $\boldsymbol{x}_t$ and consists of a set of parameters for the internal used model(s) of the system at time t. The task is to estimate the state vector $\boldsymbol{x}_t$ from noisy observations $\boldsymbol{z}_{1:t}$ gathered by sensors at discrete time steps. In robotic applications, for example, the pose of the robot and the parameters of its actuators are important to fulfill complex jobs and therefore are described by internal models and are part of the state vector. In the context of this thesis, the state vector for the camera tracking task contains the varying parameters of the camera.

The subscript $1:t$ denotes the set of observations, gathered between time 1 and time t, or more generally:

$$\boldsymbol{a}_{t_1:t_2} = \{\boldsymbol{a}_t | t_1 \leq t \leq t_2\}$$

Variables without a subscript, such as $\boldsymbol{x}$ or $\boldsymbol{z}$, stand for arbitrary timepoints, the whole data set or the true system state, depending on the context.

The estimation process is called *smoothing*, if the time of interest for the system state lies in the past (i.e. newer measurements are available), *filtering* if the time of interest coincides with the time of the latest measurement and *prediction*, if an estimate for a future time point is determined. This thesis will focus on *recursive filtering*.

Static systems have unknown but constant state vectors $\boldsymbol{x}_t = \boldsymbol{x}_{t-1}$, whereas dynamic systems change their state over time according to some possibly nonlinear function g, called *system process* or *motion model*. As most systems are non-passive, they interact with the environment through a *control process* and influence the system state. This interaction is described by the l-dimensional *control vector* $\boldsymbol{u}_t$ and is considered in the system process. Even if the control vectors are obtained by sensor readings, they are not treated in the same ways as regular measurements, since they express the effects of a control action.

In the context of self-localization in robotics for example, $\boldsymbol{u}_t$ is usually the odometry reading, which reflects the relative motion of the robot from time $t-1$ to time t. In the context of this thesis, the control vector represents the relative motion of the camera between two subsequent frames.

It is obvious that the system process is also affected by noise. Therefore, state estimation has to deal with *process noise* $\boldsymbol{v}$ and *observation noise* $\boldsymbol{w}$. Assuming an accurate system model (including the sensors), the noise is not caused by systematic errors and therefore has zero mean and is independent of the state vector.

$$\begin{aligned} E[\boldsymbol{v}] &= \mathbf{0} & E[\boldsymbol{w}] &= \mathbf{0} \\ E\left[\boldsymbol{v}\boldsymbol{v}^T\right] &= \Sigma_{vv} & E\left[\boldsymbol{w}\boldsymbol{w}^T\right] &= \Sigma_{ww} \\ E[\boldsymbol{v}|\boldsymbol{x}] &= E[\boldsymbol{v}], & E[\boldsymbol{w}|\boldsymbol{x}] &= E[\boldsymbol{w}] \end{aligned}$$

Furthermore, it is supposed that the observation and the process noise are uncorrelated.

$$E\left[\boldsymbol{v}\boldsymbol{w}^T\right] = \Sigma_{vw} = 0$$

In most applications the direct observation of the state vector is not possible. Therefore, the relation between observations and the state vector is described by some possibly nonlinear function h, the *observation model* or *measurement model*. Measurements in the image, for example, correspond to features of real-world objects, and therefore are described by the nonlinear projective mapping of the camera with respect to its parameters.

The *Markov assumption* or *complete state assumption* states that future and past data are independent, if the current state $\boldsymbol{x}_t$ is known. Thus the previous state is the best predictor for the current time step, given the control vector $\boldsymbol{u}_t$. Following the Markov assumption in both the observation and the motion model leads to following parameterization:

$$\boldsymbol{z}_t = h(\boldsymbol{x}_t, \boldsymbol{w}_t) \tag{4.1}$$

$$\boldsymbol{x}_t = g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t, \boldsymbol{v}_t) \tag{4.2}$$

$\boldsymbol{v}_t$ and $\boldsymbol{w}_t$ denote the unobservable *process noise* and *observation noise* at time t respectively. In most applications $h(\boldsymbol{x}_t, \boldsymbol{w}_t)$ and $g(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t, \boldsymbol{v}_t)$ are nonlinear. However, the linear case is extremely important, as it provides the general method for the nonlinear case and therefore will be introduced first.

If the motion model and the observation model are linear, equations (4.1) and (4.2) can be

written as follows:

$$\boldsymbol{z}_t = H\boldsymbol{x}_t + \boldsymbol{w}_t \tag{4.3}$$

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \boldsymbol{v}_t \tag{4.4}$$

where H is an $m \times n$ matrix, A an $n \times n$ matrix and B an $n \times l$ matrix.[1]

As the concrete errors $\boldsymbol{w}_t$, $\boldsymbol{v}_t$ are unknown, possibly no unique solution exists and the task becomes to determine an *optimal* estimate $\hat{\boldsymbol{x}}_t$ for time t. Since *optimal* can be interpreted in different ways, various approaches exist for estimating $\hat{\boldsymbol{x}}_t$. This chapter will introduce some important estimators and show their advantages, differences and particular similarities.

4.1 Linear Least Squares Estimator

One commonly used approach for estimating the state vector or *parameter vector* is the least squares method. This method is widely used throughout the computer vision research community [Low91]. It was first described by the German mathematician Carl Friedrich Gauß around the end of the 18th century [Sor70].

We will first introduce least squares estimation for a constant and deterministic state vector $\boldsymbol{x}$, i.e. all observations gathered at different points in time describe the same state, and will be denoted in a single observation vector $\boldsymbol{z}$. Since the error $\boldsymbol{w}$ is unobservable and displays the error in the measurements, it can be removed from the equation.

$$\begin{aligned}\Sigma_{zz} &= Var\left(H\boldsymbol{x} + \boldsymbol{w}\right) \\ &= H\Sigma_{xx}H^T + \Sigma_{ww} \\ &= \Sigma_{ww}\end{aligned}$$

The task then becomes: find an estimate $\hat{\boldsymbol{x}}$ that solves the following equation:

$$\boldsymbol{z} = H\boldsymbol{x} \tag{4.5}$$

The following cases can be distinguished:

$\operatorname{rank}(H) < n$ This occurs if fewer observations are available than are required ($m < n$). I.e. more than $m - n$ rows of H are linear dependent. In this case the equation system

[1] In many applications $\boldsymbol{z}_t$ does not contain the raw measurements and H depends also on the measurements.

is underdetermined and an infinite number of solutions exists. This ambiguity can be solved if $\boldsymbol{x}$ is supposed to be a stochastic variable and *a priori* knowledge is available.

$\operatorname{rank}(H) = n = m$ In this special case, H is a regular matrix and the unique solution is given by $\boldsymbol{x} = H^{-1}\boldsymbol{z}$. Although a unique solution exists, the estimate can diverge considerably from the true state because of noise and possible outliers in the measurements.

$\operatorname{rank}(H) = n$**,** $n < m$ In this case, the equation system is overdetermined and possibly has no solution. The problem becomes to find the (in some sense) *optimal* estimate.

This section will first introduce the latter two cases, then extend the method to stochastic quantities with given *a priori* knowledge to also handle the first case. Finally, a recursive least squares estimator for linear dynamic systems will be introduced and compared with different approaches.

4.1.1 Ordinary Least Squares

$$\operatorname{rank}(H) = n \tag{4.6}$$

As the observations are afflicted with noise, possibly no estimate exists that solves the equation system uniquely, i.e. the residual vector for any estimator $\tilde{\boldsymbol{x}}$ will have a length ≥ 0.

$$\boldsymbol{r} = H\tilde{\boldsymbol{x}} - \boldsymbol{z} \tag{4.7}$$

$$\boldsymbol{r}^T\boldsymbol{r} \geq 0 \tag{4.8}$$

Note that $\boldsymbol{r}$ is called *residual* and not *error*, since the error is defined to be the difference between the measurement and the true (expected) value, whereas the residual is defined to be the difference between the measurement and the estimated value. Nevertheless, we will use the term *sum of squared errors* (SSE) instead of *sum of squared residuals* to coincide with popular literature.

The ordinary least squares estimator (*OLS*) finds the estimate $\hat{\boldsymbol{x}}$ by minimizing the sum of squared errors (*SSE*) (4.8).

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

with

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{2}\boldsymbol{r}^T\boldsymbol{r} \\ &= \frac{1}{2}(H\boldsymbol{x}-\boldsymbol{z})^T(H\boldsymbol{x}-\boldsymbol{z}) \end{aligned} \tag{4.9}$$

This function is called *cost function* or *error function*[2]. The parameter vector $\hat{\boldsymbol{x}}$ of the cost function, which gives the smallest cost value, is called *minimizer*, whereas the corresponding cost value is called *minimum*.
Necessary and sufficient conditions for a minimizer of f are given by:

$$\frac{\partial f}{\partial \boldsymbol{x}}(\hat{\boldsymbol{x}}) = \mathbf{0} \tag{4.10}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2}(\hat{\boldsymbol{x}}) > 0 \tag{4.11}$$

where equation (4.11) states that the second derivative of f, which is a symmetric matrix, has to be positive definite (see section 4.2).

$$\frac{\partial f}{\partial \boldsymbol{x}} = H^T(H\boldsymbol{x}-\boldsymbol{z}) \tag{4.12}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{x}^2} = H^T H \tag{4.13}$$

From equations (4.10) and (4.12) the *normal equation* can be derived as:

$$H^T H\boldsymbol{x} = H^T\boldsymbol{z} \tag{4.14}$$

From equations (4.13) and (4.6), it can be seen that $H^T H$ is positive definite, thus the unique solution is a minimizer of f and is given by:

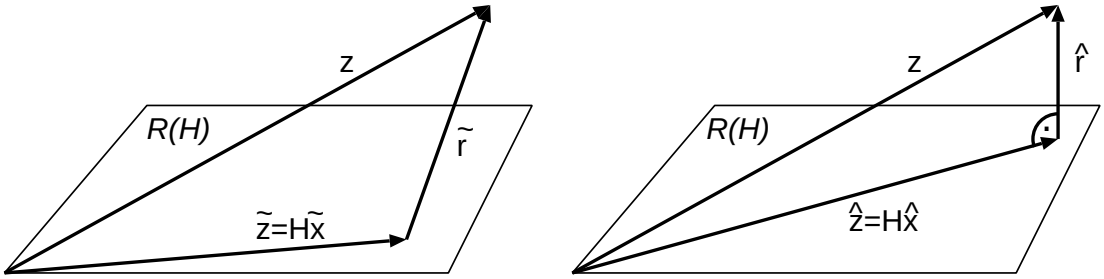
$$\hat{\boldsymbol{x}} = \left(H^T H\right)^{-1} H^T\boldsymbol{z} \tag{4.15}$$

The matrix $\left(H^T H\right)^{-1} H^T = H^\dagger$ is called *pseudoinverse* of H. Note, that $H^\dagger H = I_n$, but $HH^\dagger \neq I_m$ in general.

[2]All these terms are interchangeable and will be used alternated throughout this thesis

Geometric interpretation of OLS

Let $R(H)$ be the column space of H. Then $R(H)$ is a n dimensional subspace of the m dimensional Euclidean space $\mathbb{R}^m$. For each estimator $\tilde{\boldsymbol{x}}$, the vector $\tilde{\boldsymbol{z}} = H\tilde{\boldsymbol{x}}$ is the linear combination of the column vectors of H and lies in $R(H)$, whereas the measurement vector $\boldsymbol{z}$ lies in $\mathbb{R}^m$. For the special case where $m = n$ or $\boldsymbol{w} = \mathbf{0}$ (noise-free data), $\boldsymbol{z}$ lies also in $R(H)$ and there exists a linear combination $\hat{\boldsymbol{x}}$ of the column vectors uniquely describing $\boldsymbol{z}$, i.e. $\boldsymbol{z} = H\hat{\boldsymbol{x}}$. However, in the general case $\boldsymbol{z}$ does not lie in $R(H)$ and therefore can only be described by $\tilde{\boldsymbol{x}}$ and an additional m-dimensional residual vector $\tilde{\boldsymbol{r}}$, i.e. $\boldsymbol{z} = H\tilde{\boldsymbol{x}} + \tilde{\boldsymbol{r}}$ (see figure 4.1(a)).



(a) (b)

Figure 4.1 Geometric interpretation of OLS: (a) arbitrary estimate with corresponding residual, and (b) OLS estimate with smallest residual

The OLS estimate $\hat{\boldsymbol{x}}$ is the vector with the smallest difference to the measurement vector $\boldsymbol{z}$, i.e. the length of the residual vector $\boldsymbol{r}$ is minimal. The residual vector with minimal length $\hat{\boldsymbol{r}}$ is perpendicular on $R(H)$ (see figure 4.1(b)), thus the scalar product of all columns of H with $\hat{\boldsymbol{r}}$ is 0. Then the solution is given by:

$$\begin{aligned} H^T\hat{\boldsymbol{r}} &= \mathbf{0} \\ H^T(H\hat{\boldsymbol{x}} - \boldsymbol{z}) &= \mathbf{0} \\ H^T H\hat{\boldsymbol{x}} &= H^T\boldsymbol{z} \end{aligned}$$

which results again in the normal equation (4.14).

4.1.2 Best Linear Unbiased Estimators

This section will introduce the class of linear unbiased estimators and in particular best linear unbiased estimators (*BLUE*).

The class of **linear** estimators are given by:

$$\tilde{\boldsymbol{x}} = A\boldsymbol{z} + \boldsymbol{a} \tag{4.16}$$

All **unbiased** linear estimators have to conform:

$$\begin{aligned}\boldsymbol{x} &= E\left[\tilde{\boldsymbol{x}}\right] \\ &= E\left[A\boldsymbol{z} + \boldsymbol{a}\right] \\ &= E\left[A\left(H\boldsymbol{x} + \boldsymbol{w}\right) + \boldsymbol{a}\right] \\ &= A\left(HE\left[\boldsymbol{x}\right] + E\left[\boldsymbol{w}\right]\right) + E\left[\boldsymbol{a}\right] \\ &= AH\boldsymbol{x} + \boldsymbol{a}\end{aligned} \tag{4.17}$$

Since the matrix A has to be independent of $\boldsymbol{z}$ [Bar74], all **linear unbiased** estimators have to satisfy following conditions:

$$AH = I_n \tag{4.18}$$

$$\boldsymbol{a} = \boldsymbol{0} \tag{4.19}$$

equation (4.16) then becomes:

$$\tilde{\boldsymbol{x}} = A\boldsymbol{z} \tag{4.20}$$

with the following covariance matrix (see section 2.3.1.1):

$$\Sigma_{\tilde{x}\tilde{x}} = A\Sigma_{zz}A^T \tag{4.21}$$

The best linear unbiased estimator (*BLUE*) $\hat{\boldsymbol{x}}$ is the estimate in the class of linear unbiased estimators with minimum variance.

$$\Sigma_{\tilde{x}\tilde{x}} \geq \Sigma_{\hat{x}\hat{x}} \tag{4.22}$$

As the variance of the estimate depends on the variance of the measurement, we will distinguish the two cases described below.

Homoscedastic Case

If the components of the observation vector are independent and identical distributed (*iid* or *homoscedastic*), or the covariance matrix Σ_{zz} is unknown and it is reasonable to assume iid, Σ_{zz} has the following structure:

$$\Sigma_{zz} = \Sigma_{ww} = \sigma_w^2 I_m \tag{4.23}$$

$$\sigma_w^2 > 0 \tag{4.24}$$

In this case, the *Gauß-Markov* theorem states [Wer97, KS79, Koc00], that the best linear unbiased estimator (*BLUE*) is given by the OLS solution (4.15).
The covariance matrix of $\hat{\boldsymbol{x}}$ is given by:

$$\begin{aligned}\Sigma_{\hat{x}\hat{x}} &= \left(H^T H\right)^{-1} H^T \Sigma_{zz} H \left(H^T H\right)^{-T} \\ &= \left(H^T H\right)^{-1} H^T \left(\sigma_w^2 I_m\right) H \left(H^T H\right)^{-1} \\ &= \sigma_w^2 \left(H^T H\right)^{-1}\end{aligned} \tag{4.25}$$

It can easily be shown that the estimator is unbiased.

$$\begin{aligned}E\left[\hat{\boldsymbol{x}}\right] &= E\left[\left(H^T H\right)^{-1} H^T \boldsymbol{z}\right] \\ &= E\left[\left(H^T H\right)^{-1} H^T \left(H\boldsymbol{x} + \boldsymbol{w}\right)\right] \\ &= \boldsymbol{x} + \left(H^T H\right)^{-1} H^T E\left[\boldsymbol{w}\right] \\ &= \boldsymbol{x}\end{aligned} \tag{4.26}$$

The BLUE (minimum variance) property of the OLS estimator can be shown as follows:
Let $\tilde{\boldsymbol{x}}$ be an arbitrary linear unbiased estimator given by equation (4.20). Its covariance matrix according to (4.21) and (4.23) is given by:

$$\begin{aligned}\Sigma_{\tilde{x}\tilde{x}} &= A\left(\sigma_w^2 I_m\right) A^T \\ &= \sigma_w^2 A A^T\end{aligned} \tag{4.27}$$

Let D be the difference between A and $\left(H^T H\right)^{-1} H^T$

$$\begin{aligned}D &= A - \left(H^T H\right)^{-1} H^T \\ A &= D + \left(H^T H\right)^{-1} H^T\end{aligned} \tag{4.28}$$

Then, the covariance matrix for the estimate $\tilde{\boldsymbol{x}}$ is given by equation (4.27).

$$\begin{aligned}
\Sigma_{\tilde{x}\tilde{x}} &= \sigma_w^2 AA^T \\
AA^T &= \left(D + \left(H^T H\right)^{-1} H^T\right)\left(D + \left(H^T H\right)^{-1} H^T\right)^T \\
&= DD^T + \left(H^T H\right)^{-1} H^T D^T + DH\left(H^T H\right)^{-1} + \left(H^T H\right)^{-1} \\
&= DD^T + \left(H^T H\right)^{-1}
\end{aligned} \tag{4.29}$$

since $DH = 0$, which can be shown by using equation (4.18)

$$\begin{aligned}
DH &= \left(A - \left(H^T H\right)^{-1} H^T\right) H \\
&= AH - \left(H^T H\right)^{-1} H^T H \\
&= AH - I_n = 0
\end{aligned} \tag{4.30}$$

As the matrix DD^T is positive semidefinite by construction, the covariance matrix of any other linear unbiased estimator is greater. And the estimator with least variance is given if $D = 0$, thus by the OLS estimate. A more intuitive way of revealing the least variance property of the OLS estimate is to show that for all $D \neq 0$:

1. there is no direction with smaller variance:

$$\forall \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{y}^T\boldsymbol{y} \neq 0 : \boldsymbol{y}^T \Sigma_{\tilde{x}\tilde{x}} \boldsymbol{y} \geq \boldsymbol{y}^T \Sigma_{\hat{x}\hat{x}} \boldsymbol{y} \tag{4.31}$$

2. there exists at least one direction with higher variance:

$$\exists \boldsymbol{y} \in \mathbb{R}^n, \boldsymbol{y}^T\boldsymbol{y} \neq 0 : \boldsymbol{y}^T \Sigma_{\tilde{x}\tilde{x}} \boldsymbol{y} > \boldsymbol{y}^T \Sigma_{\hat{x}\hat{x}} \boldsymbol{y} \tag{4.32}$$

To show (4.31) we apply equation (4.29) to equation (2.23), and by considering (4.24) we get:

$$\begin{aligned}
\boldsymbol{y}^T \Sigma_{\tilde{x}\tilde{x}} \boldsymbol{y} &\geq \boldsymbol{y}^T \Sigma_{\hat{x}\hat{x}} \boldsymbol{y} \\
\boldsymbol{y}^T \left(\Sigma_{\tilde{x}\tilde{x}} - \Sigma_{\hat{x}\hat{x}}\right) \boldsymbol{y} &\geq 0 \\
\boldsymbol{y}^T \left(\sigma_w^2 \left(DD^T + \left(H^T H\right)^{-1}\right) - \sigma_w^2 \left(H^T H\right)^{-1}\right) \boldsymbol{y} &\geq 0 \\
\boldsymbol{y}^T \left(DD^T\right) \boldsymbol{y} &\geq 0
\end{aligned}$$

which is true $\forall \boldsymbol{y} : \boldsymbol{y}^T\boldsymbol{y} \neq 0$, as DD^T is positive semidefinite by construction. No direction with smaller variance exists.

To show (4.32), we look at the diagonal elements (c_{ii}) of DD^T, which are defined as

$$c_{ii} = \sum_{j=1}^{m} d_{ij}^2 \qquad 1 \leq i \leq n \tag{4.33}$$

with $D = (d_{ij})$. Thus, for all $D \neq 0$, at least one diagonal element is greater than zero.

$$\forall D : (D \neq \mathbf{0} \Leftrightarrow \exists i : 1 \leq i \leq n \wedge c_{ii} > 0) \tag{4.34}$$

Without loss of generality, we suppose, that $c_{11} > 0$. Then, at least the vector $y = (1, 0, \ldots, 0)^T$ has greater variance.

Estimating the Unknown Measurement Variance

If the measurement variance σ_w^2 is unknown, it can be estimated using the following unbiased estimator:

$$\hat{\sigma}_w^2 = \frac{(\boldsymbol{z} - H\hat{\boldsymbol{x}})^T (\boldsymbol{z} - H\hat{\boldsymbol{x}})}{m - n} \tag{4.35}$$

Proof: Let $G = I_m - H\left(H^T H\right)^{-1} H^T$ with following properties:
G is symmetric:

$$G^T = \left(I_m - H\left(H^T H\right)^{-1} H^T\right)^T = I_m - H\left(H^T H\right)^{-1} H^T = G \tag{4.36}$$

G is idempotent:

$$\begin{aligned} G^T G &= \left(I_m - H\left(H^T H\right)^{-1} H^T\right)\left(I_m - H\left(H^T H\right)^{-1} H^T\right) \\ &= I_m - 2H\left(H^T H\right)^{-1} H^T + H\underbrace{\left(H^T H\right)^{-1} H^T H}_{I_n}\left(H^T H\right)^{-1} H^T \\ &= \left(I_m - H\left(H^T H\right)^{-1} H^T\right) = G \end{aligned} \tag{4.37}$$

and the trace of G satisfies:

$$\begin{aligned} tr\,(G) &= tr\,(I_m) - tr\left(H\left(H^T H\right)^{-1} H^T\right) \\ &= tr\,(I_m) - tr\left(H^T H\left(H^T H\right)^{-1}\right) \\ &= tr\,(I_m) - tr\,(I_n) = m - n \end{aligned} \tag{4.38}$$

Then

$$\begin{aligned} \boldsymbol{z} - H\hat{\boldsymbol{x}} &= \boldsymbol{z} - H\left(H^T H\right)^{-1} H^T \boldsymbol{z} \\ &= G\boldsymbol{z} = G\left(H\boldsymbol{x} + \boldsymbol{w}\right) \\ &= G\boldsymbol{w} \end{aligned} \tag{4.39}$$

since $GH = 0$

$$\begin{aligned} GH &= \left(I_m - H\left(H^T H\right)^{-1} H^T\right) H \\ &= H - H = 0 \end{aligned} \tag{4.40}$$

The expected value $E\left[\hat{\sigma}_w^2\right]$ then becomes:

$$\begin{aligned} E\left[\hat{\sigma}_w^2\right] &= \frac{1}{m-n} E\left[\boldsymbol{w}^T G^T G \boldsymbol{w}\right] \\ &= \frac{1}{m-n} E\left[\boldsymbol{w}^T G \boldsymbol{w}\right] \\ &= \frac{1}{m-n} \mathrm{tr}\left(E\left[G\boldsymbol{w}\boldsymbol{w}^T\right]\right) \\ &= \frac{1}{m-n} \mathrm{tr}\left(GE\left[\boldsymbol{w}\boldsymbol{w}^T\right]\right) \\ &= \frac{1}{m-n} \mathrm{tr}\left(G\sigma_w^2 I_m\right) \\ &= \frac{\sigma_w^2}{m-n} tr\left(G\right) = \sigma_w^2 \end{aligned} \tag{4.41}$$

The unbiased covariance matrix $\hat{\Sigma}_{\hat{x}\hat{x}}$ of the estimate $\hat{\boldsymbol{x}}$ is given by:

$$\hat{\Sigma}_{\hat{x}\hat{x}} = \hat{\sigma}_w^2 \left(H^T H\right)^{-1} \tag{4.42}$$

Note: for $n = 1$ the OLS estimate results in the sample mean and $\hat{\sigma}_w^2$ results in the sample variance.

General Least Squares

In multisensor systems some sensors usually provide more accurate measurements than others and measurements can be autocorrelated. For example, laser measurements are more accurate than sonar measurements and measurements in images have usually a strong correlation between the x and y value. This leads to a covariance matrix which does not satisfy the above

condition (4.23), and the OLS solution is not BLUE.

It is obvious that multiplying an arbitrary $k \times m$ matrix W with $\operatorname{rank}(W) = m$ on both sides of the equation system (4.5) does not violate the equation, but changes its OLS solution.

$$\begin{aligned} \underbrace{W\boldsymbol{z}}_{\boldsymbol{z}'} &= \underbrace{WH}_{H'}\boldsymbol{x} \\ \hat{\boldsymbol{x}}' &= \left(H'^T H'\right)^{-1} H'^T \boldsymbol{z}' \end{aligned} \tag{4.43}$$

The OLS solution remains unbiased.

$$\begin{aligned} E\left[\hat{\boldsymbol{x}}'\right] &= \left(H'^T H'\right)^{-1} H'^T E\left[\boldsymbol{z}'\right] \\ &= \left(H^T W^T W H\right)^{-1} H^T W^T W H E\left[\boldsymbol{x}\right] \\ &= E\left[\boldsymbol{x}\right] = \boldsymbol{x} \end{aligned} \tag{4.44}$$

with its covariance matrix:

$$\Sigma_{\hat{x}'\hat{x}'} = \left(H^T W^T W H\right)^{-1} H^T W^T W \Sigma_{zz} W^T W H \left(H^T W^T W H\right)^{-1} \tag{4.45}$$

Known Covariance Matrix

This leads to the following consideration: If the covariance matrix is known or at least its structure up to a constant multiplier σ_w^2 is known

$$\tilde{\Sigma}_{zz} = \sigma_w^2 \Sigma_{zz} \tag{4.46}$$

the equation system can be transformed so that $\boldsymbol{z}'$ becomes iid, by selecting a suitable weight matrix. Let C be the square root of the covariance matrix with $\tilde{\Sigma}_{zz} = CC^T$, then the weight matrix is selected as $W = C^{-1}$. The transformed observation vector becomes:

$$\boldsymbol{z}' = W\boldsymbol{z} = C^{-1}\boldsymbol{z} \tag{4.47}$$

with its covariance matrix:

$$\begin{aligned} \Sigma_{z'z'} &= C^{-1}\Sigma_{zz}C^{-T} \\ &= C^{-1}\left(\sigma_w^2 CC^T\right)C^{-T} \\ &= \sigma_w^2 I_m \end{aligned} \tag{4.48}$$

Then the OLS estimate of the transformed equation system becomes BLUE [FHT96, Str90, Lue91, Wer97], and is given by:

$$\begin{aligned}\hat{\boldsymbol{x}} &= \left(H'^T H'\right)^{-1} H'^T \boldsymbol{z}' \\ &= \left(H^T C^{-T} C^{-1} H\right)^{-1} H^T C^{-T} C^{-1} \boldsymbol{z} \\ &= \left(H^T \tilde{\Sigma}_{zz}^{-1} H\right)^{-1} H^T \tilde{\Sigma}_{zz}^{-1} \boldsymbol{z}\end{aligned} \tag{4.49}$$

Its covariance matrix is given by:

$$\Sigma_{\hat{x}\hat{x}} = \left(H^T \tilde{\Sigma}_{zz}^{-1} H\right)^{-1} \tag{4.50}$$

Unlike the OLS estimate, which minimizes the squared Euclidean distance $\boldsymbol{r}^T\boldsymbol{r}$, general least squares minimizes the squared Σ_{zz}^{-1} norm $\boldsymbol{r}^T\Sigma_{zz}^{-1}\boldsymbol{r}$ (squared Mahalanobis distance of the residual). This estimator is also called *Gauss-Markov Estimator*, and a detailed discussion of this estimator can be found in [BS89, BSF88].
The special case where the measurement errors are independent but not identically distributed, i.e. $\Sigma_{zz} = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_m^2)$, is called *Weighted Least Squares* (WLS).

Estimating the Measurement Variance

If the measurement variance σ_w^2 is unknown, its unbiased estimator is given by:

$$\hat{\sigma}_w^2 = \frac{(\boldsymbol{z} - H\hat{\boldsymbol{x}})^T \tilde{\Sigma}_{zz}^{-1} (\boldsymbol{z} - H\hat{\boldsymbol{x}})}{m - n} \tag{4.51}$$

4.1.3 Least Squares Estimation with Given Prior Knowledge

If prior knowledge (or *a priori* knowledge), about the system state is given by some linear equation:

$$\tilde{\boldsymbol{z}} = B\boldsymbol{x} + \tilde{\boldsymbol{w}} \tag{4.52}$$

where B is supposed to have full rank and $\tilde{\boldsymbol{w}}$ is a random variable with zero-mean and covariance matrix $\Sigma_{\tilde{w}\tilde{w}}$, denoting the noise or accuracy of the prior information.
According to the notation of equation (4.5) it can be rewritten as

$$\tilde{\boldsymbol{z}} = B\boldsymbol{x} \tag{4.53}$$

with $\Sigma_{\tilde{z}\tilde{z}} = \Sigma_{\tilde{w}\tilde{w}}$.
The *Minimum Variance Estimator* [BSF88, BS89] can be determined by composing the measurement equation (4.5) with the prior knowledge to a new equation system.

$$\boldsymbol{y} = D\boldsymbol{x} \tag{4.54}$$

with

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{z} \\ \tilde{\boldsymbol{z}} \end{pmatrix} \qquad D = \begin{pmatrix} H \\ B \end{pmatrix} \qquad \Sigma_{yy} = \begin{pmatrix} \Sigma_{zz} & \Sigma_{z\tilde{z}} \\ \Sigma_{\tilde{z}z} & \Sigma_{\tilde{z}\tilde{z}} \end{pmatrix}$$

Assuming that the prior knowledge and the new measurement are uncorrelated leads to $\Sigma_{z\tilde{z}} = \Sigma_{\tilde{z}z}^T = 0$, and the minimum variance estimate is given by:

$$\hat{\boldsymbol{x}} = \left(H^T\Sigma_{zz}^{-1}H + B^T\Sigma_{\tilde{z}\tilde{z}}^{-1}B\right)^{-1}\left(H^T\Sigma_{zz}^{-1}\boldsymbol{z} + B^T\Sigma_{\tilde{z}\tilde{z}}^{-1}\tilde{\boldsymbol{z}}\right) \tag{4.55}$$

$$\Sigma_{\hat{x}\hat{x}} = \left(H^T\Sigma_{zz}^{-1}H + B^T\Sigma_{\tilde{z}\tilde{z}}^{-1}B\right)^{-1} \tag{4.56}$$

Usually the prior knowledge is given directly by $\tilde{\boldsymbol{x}}$ with covariance $\Sigma_{\tilde{x}\tilde{x}}$, and is determined from a *prediction* step; thus we will call it *prediction* in the following. In this case, the minimum variance estimate becomes:

$$\hat{\boldsymbol{x}} = \left(H^T\Sigma_{zz}^{-1}H + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right)^{-1}\left(H^T\Sigma_{zz}^{-1}\boldsymbol{z} + \Sigma_{\tilde{x}\tilde{x}}^{-1}\tilde{\boldsymbol{x}}\right) \tag{4.57}$$

$$\Sigma_{\hat{x}\hat{x}} = \left(H^T\Sigma_{zz}^{-1}H + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right)^{-1} \tag{4.58}$$

A useful side effect of the latter case is, that, regardless of the number of observations made, the composed equation system is never underdetermined, and therefore an optimal solution can always be obtained. This estimator is discussed in more detail in ([BS89, BSF88]). It can be seen that if $\Sigma_{\tilde{x}\tilde{x}} \to \infty$, i.e. the *a priori* knowledge is *non-informative*, we get the *Gauss-Markov* estimator as described in section 4.1.2.

4.1.4 Recursive Least Squares Estimation

Almost all applications for time-varying system states, and some applications for constant system states require an estimate in each time step. In the context of this thesis, estimating the camera parameters for each frame is important for further processing. E.g. determining the player and ball trajectories.

Regardless of dynamic or static systems, storing all observations up to the current time

step and determining the best estimate from all these observations for each time step can be very inefficient. Both the computational and memory inefficiency can be resolved by using a recursive method, which provides a sequential rather than a batch processing. The clou of recursive methods is that they update an estimate for newly available measurements without the need to store and reprocess all past measurements.

4.1.4.1 Static Systems

In the previous section we considered *a priori* information of the form:

$$B\boldsymbol{x} = \tilde{\boldsymbol{z}}$$

For a static system, this equation can be interpreted as the composition of all measurements gathered in the past (up to time $t-1$).

$$B = \left(H_1^T, H_2^T, \ldots, H_{t-1}^T\right)^T \tag{4.59}$$

$$\tilde{\boldsymbol{z}} = \left(\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, \ldots, \boldsymbol{z}_{t-1}^T\right)^T \tag{4.60}$$

$$\Sigma_{\tilde{z}\tilde{z}} = \operatorname{diag}\left(\Sigma_{z_1 z_1}, \Sigma_{z_2 z_2}, \ldots, \Sigma_{z_{t-1} z_{t-1}}\right) \tag{4.61}$$

whereas equation (4.5) presents the current measurements for time t (i.e. $\boldsymbol{z} = \boldsymbol{z}_t$).

However, matrix B, vector $\tilde{\boldsymbol{z}}$ and its covariance matrix $\Sigma_{\tilde{z}\tilde{z}}$ can get very large for long time periods, and the latter case (4.57) is more favorable as only $\tilde{\boldsymbol{x}}$ and $\Sigma_{\tilde{x}\tilde{x}}$ are stored. Both $\tilde{\boldsymbol{x}}$ and $\Sigma_{\tilde{x}\tilde{x}}$ can be determined from equation (4.52) by the Gauß-Markov estimate.

$$\tilde{\boldsymbol{x}} = \left(B^T \Sigma_{\tilde{z}\tilde{z}}^{-1} B\right)^{-1} B^T \Sigma_{\tilde{z}\tilde{z}}^{-1} \tilde{\boldsymbol{z}} \tag{4.62}$$

$$\Sigma_{\tilde{x}\tilde{x}} = \left(B^T \Sigma_{\tilde{z}\tilde{z}}^{-1} B\right)^{-1} \tag{4.63}$$

Using this result in equations (4.57) and (4.58) again delivers (4.55) and (4.56) respectively, and the recursive minimum variance estimator for static systems can easily be derived as:

$$\tilde{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t-1} \tag{4.64}$$

$$\tilde{P}_t = \hat{P}_{t-1} \tag{4.65}$$

$$\hat{\boldsymbol{x}}_t = \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} \boldsymbol{z}_t + \tilde{P}_t^{-1} \tilde{\boldsymbol{x}}_t\right) \tag{4.66}$$

$$\hat{P}_t = \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \tag{4.67}$$

To avoid double indexing, the following substitutions have been made:

$$\begin{aligned}\Sigma_{\hat{x}\hat{x}} &=: \hat{P} \\ \Sigma_{\tilde{x}\tilde{x}} &=: \tilde{P} \\ \Sigma_{zz} = \Sigma_{ww} &=: Q\end{aligned}$$

By converting equation (4.66) to

$$\hat{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_t + K_t \left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right) \tag{4.68}$$

we get a more intuitive representation. The $n \times m$ weight matrix K_t is called *Kalman Gain* and can be derived from equation (4.66) as follows:

$$\begin{aligned}\hat{\boldsymbol{x}}_t &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} \boldsymbol{z} + \tilde{P}_t^{-1} \tilde{\boldsymbol{x}}_t\right) \\ &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} \boldsymbol{z}_t + \tilde{P}_t^{-1} \tilde{\boldsymbol{x}}_t + H_t^T Q_t^{-1} H_t \tilde{\boldsymbol{x}}_t - H_t^T Q_t^{-1} H_t \tilde{\boldsymbol{x}}_t\right) \\ &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(\left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right) \tilde{\boldsymbol{x}}_t + H_t^T Q_t^{-1} \left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right)\right) \\ &= \tilde{\boldsymbol{x}}_t + \underbrace{\left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} H_t^T Q_t^{-1}}_{K_t} \left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right)\end{aligned}$$

The residual $\left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right)$ in the above equation (4.68) is called *innovation*, and reflects the difference between the current measurement $\boldsymbol{z}_t$ and its prediction $H_t \tilde{\boldsymbol{x}}_t$. The Kalman gain is used to get the weighted average between the predicted state $\tilde{\boldsymbol{x}}_t$ and the innovation, which minimizes the *a posteriori* error covariance.

A more favorable representation of the Kalman gain with only one matrix inversion can be derived as follows:

$$\begin{aligned}K_t &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} H_t^T Q_t^{-1} \\ &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} H_t^T Q_t^{-1} \left(H_t \tilde{P}_t H_t^T + Q_t\right) \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1} \\ &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} H_t \tilde{P}_t H_t^T + H_t^T\right) \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1} \\ &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right) \tilde{P}_t H_t^T \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1} \\ &= \tilde{P}_t H_t^T \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1}\end{aligned}$$

The covariance matrix of the estimate (4.67) can also be specified in terms of the Kalman gain.

$$\begin{aligned}
\hat{P}_t &= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \\
&= \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \tilde{P}_t^{-1} \tilde{P}_t \\
&= \left(I_n - I_n + \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \tilde{P}_t^{-1}\right) \tilde{P}_t \\
&= \left(I_n - \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right) + \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} \tilde{P}_t^{-1}\right) \tilde{P}_t \\
&= \left(I_n - \left(H_t^T Q_t^{-1} H_t + \tilde{P}_t^{-1}\right)^{-1} H_t^T Q_t^{-1} H_t\right) \tilde{P}_t \\
&= \left(I_n - K_t H_t\right) \tilde{P}_t
\end{aligned} \tag{4.69}$$

Using the Kalman gain K_t, the recursive minimum variance estimator for static systems can be rewritten to:

$$\tilde{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t-1} \tag{4.70}$$

$$\tilde{P}_t = \hat{P}_{t-1} \tag{4.71}$$

$$K_t = \tilde{P}_t H_t^T \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1} \tag{4.72}$$

$$\hat{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_t + K_t \left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right) \tag{4.73}$$

$$\hat{P}_t = \left(I_n - K_t H_t\right) \tilde{P}_t \tag{4.74}$$

4.1.4.2 Linear Systems

The recursive form of the *Minimum Variance* estimator for linear process and observation models is given by the following equations:

$$\tilde{\boldsymbol{x}}_t = A_t \hat{\boldsymbol{x}}_{t-1} + B_t \boldsymbol{u}_t \tag{4.75}$$

$$\tilde{P}_t = A_t \hat{P}_{t-1} A_t^T + R_t \tag{4.76}$$

$$K_t = \tilde{P}_t H_t^T \left(H_t \tilde{P}_t H_t^T + Q_t\right)^{-1} \tag{4.77}$$

$$\hat{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}}_t + K_t \left(\boldsymbol{z}_t - H_t \tilde{\boldsymbol{x}}_t\right) \tag{4.78}$$

$$\hat{P}_t = \left(I_n - K_t H_t\right) \tilde{P}_t \tag{4.79}$$

To avoid double indexing, we substitute Σ_{vv} by R.

Equations (4.75) and (4.76) are called the *prediction* or *time update stage*, and equations

(4.77) - (4.79) are called *update* or *measurement update stage*. The method differs only in the time update stage from the static case. As can be seen, the prediction is determined by applying the system process (4.4) to the previous estimate $\hat{\boldsymbol{x}}_{t-1}$ using the current control vector $\boldsymbol{u}_t$.

These equations are equivalent to the well known *Kalman Filter*, and a detailed derivation (based on least squares) of these equations can be found in [BSF88, BS89, Fau93, BP99, Sor80].

At time $t = 1$, *a priori* information $\hat{\boldsymbol{x}}_0$ and $\hat{P}_0$ is required. If no prior knowledge is available, the first reliable estimate can be made after n linear independent observations and is the recursive form of the Gauss-Markov estimator.

The Kalman Filter can also be derived as the Gaussian case of the *Bayes Filter* and will be discussed in section 4.5.

4.2 Non-Linear Least Squares Estimation

So far we have discussed state estimation based on linear motion models and observation models. In real-world applications, however, these assumptions are hardly true and linear estimation methods are infeasible.

This section will introduce the problem of finding the least squares solution of nonlinear observation models, which is a special variant of the more general *nonlinear optimization* problem: Given a function $f : \mathbb{R}^m \mapsto \mathbb{R}$, find an estimate that minimizes the value of this so-called *objective function* or *cost function*.

$$\boldsymbol{x}^+ = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$$

with $\boldsymbol{x}^+$ being the *global minimizer* of f.
Least squares estimates consider *cost functions* of the following form:

$$\begin{aligned} f(\boldsymbol{x}) &= \frac{1}{2} \left(h(\boldsymbol{x}) - \boldsymbol{z}\right)^T \left(h(\boldsymbol{x}) - \boldsymbol{z}\right) \\ &= \frac{1}{2} \boldsymbol{r}^T(\boldsymbol{x})\, \boldsymbol{r}(\boldsymbol{x}) \end{aligned} \tag{4.80}$$

The problem of finding the global minimizer $\boldsymbol{x}^+$ of f is very hard to solve in general. Therefore only methods for finding a *local minimizer* $\boldsymbol{x}^*$ will be presented in this section. Since a closed form solution, unlike in the linear case, in general does not exist or is hard to determine, all nonlinear methods presented in this section are iterated starting at an initial estimate $\boldsymbol{x}_0$. In each iteration k the estimate is refined by moving along a descending direction $\boldsymbol{l}_k$ with a step

width ρ_k.

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \rho_k \boldsymbol{l}_k \tag{4.81}$$

A sequence of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ is produced, which in most cases converges to $\boldsymbol{x}^*$.

$$\boldsymbol{x}_k \rightarrow \boldsymbol{x}^* \qquad\qquad k \rightarrow \infty \tag{4.82}$$

The iteration is stopped if a *termination condition* is satisfied. Usually the termination condition is met if a sufficient accuracy in the estimated parameters or a given maximum iteration k_{max} is reached, and the estimate (iterate) from the last iteration is accepted as an approximation of $\boldsymbol{x}^*$.
If the given function has several local minimizers, the result is highly dependent on the initial point $\boldsymbol{x}_0$. In general convergence is not necessarily guaranteed, and if convergence occurs, the estimate is not necessarily the closest local minimizer to the initial point. To enforce convergence, most methods require the so-called *descending condition*:

$$f\left(\boldsymbol{x}_k\right) < f\left(\boldsymbol{x}_{k-1}\right) \tag{4.83}$$

Usually the descending condition is satisfied by selecting a suitable step size ρ_k in each iteration.

All methods discussed in this section differ only in the selection of the descending direction $\boldsymbol{l}_k$ and the step size ρ_k. Their main difference is the determination of the descending direction, which is given for all methods by:.

$$\boldsymbol{l}_k = -\tilde{H}_k^{-1} J_k^T \tag{4.84}$$

where $\tilde{H}_k$ is an approximation of the Hessian of f and J_k is the Jacobian of the residual vector $\boldsymbol{r}$ in iteration k. Therefore, the main difference between all methods is the approximation of the Hessian.

Convergence rate

The *convergence rate* indicates the expected improvement in each iteration. Methods with small convergence rates, in general, need more iterations to achieve a given accuracy in the estimated parameters than methods with a high convergence rate. We distinguish between the following convergence rates:

Linear convergence

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq a\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*\| \qquad 0 < a < 1 \tag{4.85}$$

Superlinear convergence

$$\frac{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*\|} \to 0 \qquad k \to \infty \tag{4.86}$$

Quadratic convergence

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq a\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*\|^2 \qquad 0 < a \tag{4.87}$$

with $\|\cdot\|$ being the 2-norm (or L_2 norm, Euclidean norm).

Necessary and Sufficient Conditions

The necessary and sufficient conditions for a local minimum are given by (4.10) and (4.11) respectively. Points satisfying the necessary condition (4.10) are called *stationary points*. Stationary points can be local minima, local maxima or saddle points. The type of a stationary point can be determined by the definiteness of the *Hessian*, which is a symmetric matrix containing the second partial derivatives of f. Since we are only interested in local minima, the Hessian of f has to be positive definite (4.11).

First and second derivatives

The gradient $\nabla f = J_f^T$ can be derived from equation (4.80)

$$\begin{aligned}\nabla f(\boldsymbol{x}) &= \left(\sum_{j=1}^{m} \frac{\partial r_j}{\partial x_1}(\boldsymbol{x})\, r_j(\boldsymbol{x}), \sum_{j=1}^{m} \frac{\partial r_j}{\partial x_2}(\boldsymbol{x})\, r_j(\boldsymbol{x}), \ldots, \sum_{j=1}^{m} \frac{\partial r_j}{\partial x_n}(\boldsymbol{x})\, r_j(\boldsymbol{x})\right)^T \\ &= J_r^T(\boldsymbol{x})\, \boldsymbol{r}(\boldsymbol{x})\end{aligned} \tag{4.88}$$

where the Jacobian J_r is defined as:

$$J_r(\boldsymbol{x}) = \frac{\partial \boldsymbol{r}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{x}}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial r_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial r_1}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial r_1}{\partial x_n}(\boldsymbol{x}) \\ \frac{\partial r_2}{\partial x_1}(\boldsymbol{x}) & \frac{\partial r_2}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial r_2}{\partial x_n}(\boldsymbol{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1}(\boldsymbol{x}) & \frac{\partial r_m}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial r_m}{\partial x_n}(\boldsymbol{x}) \end{pmatrix} \tag{4.89}$$

The Hessian $\nabla^2 f$ is given by:

$$\nabla^2 f = \begin{pmatrix} \sum_{i=1}^m \left(\frac{\partial^2 r_i}{\partial x_1 \partial x_1} r_i + \frac{\partial r_i}{\partial x_1} \frac{\partial r_i}{\partial x_1} \right) & \dots & \sum_{i=1}^m \left(\frac{\partial^2 r_i}{\partial x_1 \partial x_n} r_i + \frac{\partial r_i}{\partial x_1} \frac{\partial r_i}{\partial x_n} \right) \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^m \left(\frac{\partial^2 r_i}{\partial x_n \partial x_1} r_i + \frac{\partial r_i}{\partial x_n} \frac{\partial r_i}{\partial x_1} \right) & \dots & \sum_{i=1}^m \left(\frac{\partial^2 r_i}{\partial x_n \partial x_n} r_i + \frac{\partial r_i}{\partial x_n} \frac{\partial r_i}{\partial x_n} \right) \end{pmatrix} \tag{4.90}$$

$$= J_r^T(\boldsymbol{x}) J_r(\boldsymbol{x}) + \sum_{i=1}^m r_i(\boldsymbol{x}) \nabla^2 r_i(\boldsymbol{x}) \tag{4.91}$$

with $\nabla^2 r_i$ being the Hessian of function $r_i(\boldsymbol{x})$.

$$\nabla^2 r_i(\boldsymbol{x}) = \frac{\partial^2 r_i}{\partial \boldsymbol{x} \partial \boldsymbol{x}}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial^2 r_i}{\partial x_1 \partial x_1}(\boldsymbol{x}) & \dots & \frac{\partial^2 r_i}{\partial x_1 \partial x_n}(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 r_i}{\partial x_n \partial x_1}(\boldsymbol{x}) & \dots & \frac{\partial^2 r_i}{\partial x_n \partial x_n}(\boldsymbol{x}) \end{pmatrix} \tag{4.92}$$

In the following, J will denote the Jacobian of the residual J_r.

Termination Criteria

It is necessary to provide a criterion to terminate the iterative search for a local minimizer of f. Since we suppose that the produced sequence of estimates $\boldsymbol{x}_k$ converges towards a stationary point $\boldsymbol{x}^*$, the iterations should be stopped if the last estimate $\boldsymbol{x}_k$ is close to $\boldsymbol{x}^*$. However, as the stationary points are unknown, it is hard to decide whether an estimate $\boldsymbol{x}_k$ is close or far from a stationary point. Therefore, most termination criterion use following consideration. If the created sequence converges towards a stationary point, the improvement reduces from iteration to iteration, and the iterative search can be stopped if the improvement falls below a given threshold. Since the improvement of an estimate can be measured in different ways, several

termination conditions exists, and we will present the most common ones in the following.

For methods satisfying the descending condition (4.83), the iteration can be stopped if the decrease in the cost value $f(\boldsymbol{x}_k)$ gets too small and no considerable improvements of the cost value can be achieved with further iterations. The threshold for the decrease can be selected absolute or relative to the current costs.

$$|f(\boldsymbol{x}_{k-1}) - f(\boldsymbol{x}_k)| \leq \mu_1 \qquad \mu_1 > 0 \tag{4.93}$$

$$|f(\boldsymbol{x}_{k-1}) - f(\boldsymbol{x}_k)| \leq \mu_2 |f(\boldsymbol{x}_{k-1})| \qquad \mu_2 > 0 \tag{4.94}$$

The above termination criteria have the disadvantage that for flat cost functions, the parameters can change considerably while the cost value changes slightly. As we are interested in the accuracy of the parameters (minimizer) and not in the accuracy in the cost value (minimum), the above idea can be applied to the change in the parameters.

$$\|\rho_k \boldsymbol{l}_k\| \leq \delta_1 \qquad \delta_1 > 0 \tag{4.95}$$

$$\|\rho_k \boldsymbol{l}_k\| \leq \delta_2 \|\boldsymbol{x}_{k-1}\| \qquad \delta_2 > 0 \tag{4.96}$$

The most widely used termination condition can be derived from the necessary condition for a local minimum (4.10). If the current estimate $\boldsymbol{x}_k$ is sufficiently close to the local minimizer $\boldsymbol{x}^*$, the first derivative must be very close to zero and the iterations can be stopped if it is below a given threshold.

$$\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon \qquad \epsilon > 0 \tag{4.97}$$

Another termination criterion is given, if the covariance matrix of the measurements is known, and the cost function is replaced by the weighted cost function (see section 4.2.3.1). In this case, the cost value is χ^2 distributed, and the iterations can be terminated if the cost value falls below a threshold obtained by the χ^2 distribution table.

In practice, a combination of the above conditions and a maximum iteration threshold k_{max} is used.

Covariance Matrix of the Estimated Parameters

The covariance matrix of the estimated parameters for all the methods presented in this section is determined from the last iteration.

Let $\rho_{k'}\boldsymbol{l}_{k'}$ be the the correction step in the last iteration k'. The final estimate is then given by:

$$\begin{aligned}\hat{\boldsymbol{x}} &= \boldsymbol{x}_{k'} \\ &= \boldsymbol{x}_{k'-1} + \rho_{k'}\boldsymbol{l}_{k'}\end{aligned} \tag{4.98}$$

Since the quality of the final estimate does not depend on the intermediate estimates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}$, i.e. the path is not important for the quality of the local minimizer, the covariance matrix coincides with the covariance matrix of the last correction step.

$$\Sigma_{\hat{x}\hat{x}} = \rho_{k'-1}^2 \Sigma_{l_{k'-1}l_{k'-1}} \tag{4.99}$$

As the covariance matrix of the direction vector depends on the covariance matrix of the measurement vector Σ_{zz}, and if Σ_{zz} is only known up to an unknown scale σ_w, this scale can analogously to equation (4.51) be determined as:

$$\hat{\sigma}_w^2 = \frac{\left(h\left(\hat{\boldsymbol{x}}\right) - \boldsymbol{z}\right)^T \Sigma_{zz}^{-1} \left(h\left(\hat{\boldsymbol{x}}\right) - \boldsymbol{z}\right)}{m - n} \tag{4.100}$$

In the following, we will not consider this case explicitly. It is assumed that the unknown scale factor is determined and considered if necessary.

4.2.1 Gradient Descent

The *Gradient Descent* or *Steepest Descent* method is one of the simplest methods.

As the negative gradient is the direction of the steepest descent, i.e. f locally decreases fastest in the direction of the negative gradient, the *gradient descent* method selects the negative gradient of the current estimate as the descending direction $\boldsymbol{l}_k$. In the following, J_k, f_k and $\boldsymbol{r}_k$ will denote respectively the Jacobian, cost value and residual vector from the current iteration k, i.e. obtained at the previous estimate $\boldsymbol{x}_{k-1}$:

$$J_k = J\left(\boldsymbol{x}_{k-1}\right) \qquad \boldsymbol{r}_k = \boldsymbol{r}\left(\boldsymbol{x}_{k-1}\right) \qquad f_k = f\left(\boldsymbol{x}_{k-1}\right) \tag{4.101}$$

The descending direction and its covariance matrix [JEDGW81] are then given by:

$$\boldsymbol{l}_k = -\nabla f\left(\boldsymbol{x}_{k-1}\right) = -J_k^T \boldsymbol{r}_k \tag{4.102}$$

$$\Sigma_{\hat{x}\hat{x}} = J_k^T \Sigma_{zz} J_k \tag{4.103}$$

By comparing this result with equation (4.84), we can see, that the Hessian is approximated

by the identity matrix.

The selection of the step size ρ_k is extremely important for this method. Too small values for ρ_k lead to very slow convergence, whereas too large values can lead to possibly no convergence. However, a necessary condition for a valid step size ρ_k is given by equation (4.83).

Usually the step size is found by the *line-search* method [Sca85, MNO04, NW99]. The line-search method finds a step width along the descending direction, which minimizes the value of the cost function, and therefore can be described as a one-dimensional optimization problem.

$$\begin{aligned} \rho_k &= \text{lineSearch}\left(\boldsymbol{x}_{k-1}, \boldsymbol{l}_k\right) \\ &= \arg\min_{\rho>0} f\left(\boldsymbol{x}_{k-1} + \rho \boldsymbol{l}_k\right) \end{aligned} \tag{4.104}$$

Algorithm 4.1 Gradient Descent Method

Require: $\boldsymbol{x}_0$, $\epsilon > 0$, $k_{max} > 0$
1: $k \leftarrow 0$
2: **repeat**
3: $\quad k \leftarrow k + 1$
4: $\quad \boldsymbol{l}_k \leftarrow -J_k^T \boldsymbol{r}_k$
5: $\quad \rho_k \leftarrow \text{lineSearch}\left(\boldsymbol{x}_{k-1}, \boldsymbol{l}_k\right)$
6: $\quad \boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \rho_k \boldsymbol{l}_k$
7: $\quad \Sigma_k \leftarrow \rho_k^2 J_k^T \Sigma_{zz} J_k$
8: **until** $\|\boldsymbol{l}_k\| \leq \epsilon \vee k \geq k_{max}$
9: $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k$
10: $\Sigma_{\hat{x}\hat{x}} \leftarrow \Sigma_k$

In general, it is too expensive and unnecessary to determine the exact step size, and an inexact solution which achieves an adequate enhancement is preferred. Usually, a couple of step widths are evaluated and the best one is selected.

The convergence rate for this method is in general linear and therefore too slow for most vision-based applications. Algorithm 4.1 provides pseudocode for the gradient descent method.

4.2.2 Newton Method

The *Newton Method* can be derived by the necessary condition for a local minimum (4.10). The idea is to determine the new estimate $\boldsymbol{x}_k$ by linearizing the left hand side around the

previous estimate $\boldsymbol{x}_{k-1}$ and to solve the equation system for the descending direction $\boldsymbol{l}_k$.

$$\nabla f\left(\boldsymbol{x}^*\right) = \nabla f\left(\boldsymbol{x}_{k-1} + \boldsymbol{l}_k\right) = \mathbf{0} \tag{4.105}$$

The truncated Taylor expansion around the current estimate of the left hand side delivers

$$\nabla f\left(\boldsymbol{x}^*\right) \approx \nabla f\left(\boldsymbol{x}_{k-1}\right) + \nabla^2 f\left(\boldsymbol{x}_{k-1}\right) \boldsymbol{l}_k \tag{4.106}$$

and we get

$$\nabla f\left(\boldsymbol{x}_{k-1}\right) + \nabla^2 f\left(\boldsymbol{x}_{k-1}\right) \boldsymbol{l}_k = 0 \tag{4.107}$$

If $\nabla^2 f$ is positive definite, the unique solution is given by:

$$\begin{aligned} \boldsymbol{l}_k &= -\left(\nabla^2 f\left(\boldsymbol{x}_{k-1}\right)\right)^{-1} \nabla f\left(\boldsymbol{x}_{k-1}\right) \\ &= -\left(J_k^T J_k + \sum_{i=1}^{m} r_{k,i} \nabla^2 r_{k,i}\right)^{-1} J_k^T \boldsymbol{r}_k \end{aligned} \tag{4.108}$$

Usually the step size is selected as $\rho_k = 1$ and the new estimate is given by:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} - \left(\nabla^2 f\left(\boldsymbol{x}_{k-1}\right)\right)^{-1} \nabla f\left(\boldsymbol{x}_{k-1}\right) \tag{4.109}$$

with following covariance matrix [JEDGW81] of the final estimate:

$$\Sigma_{\hat{x}\hat{x}} = \left(J_k^T J_k + \sum_{i=1}^{m} r_{k,i} \cdot \nabla^2 r_{k,i}\right)^{-1} J_k^T \Sigma_{zz} J_k \left(J_k^T J_k + \sum_{i=1}^{m} r_{k,i} \cdot \nabla^2 r_{k,i}\right)^{-1} \tag{4.110}$$

Up to quadratic cost functions, the Newton method finds the exact solution in just one iteration.

The Newton method is well defined as long as the Hessian is not singular. If the Hessian is positive definite in every step and the initial point $\boldsymbol{x}_0$ is sufficiently close to the local minimizer $\boldsymbol{x}^*$, the adjustment vectors $\boldsymbol{l}_k$ are descending directions, and the method converges quadratically towards $\boldsymbol{x}^*$. Nevertheless, the classical Newton method has a couple of crucial drawbacks.

- If the initial point is far from the solution, the Hessian may not be positive definite, and the method may converge towards a maximum or saddle point.
- The Hessian may be singular or ill-conditioned and the equation system can not be solved at all or only with considerable errors in the adjustment term $\boldsymbol{l}_k$.

- Second derivatives of f are required, which makes this method inefficient for real time applications.

Algorithm 4.2 Classical Newton Method

Require: $\boldsymbol{x}_0$, $\epsilon > 0$, $k_{max} > 0$

$k \leftarrow 0$
repeat
  $k \leftarrow k + 1$
  $\boldsymbol{l}_k \leftarrow -\left(\nabla^2 f\left(\boldsymbol{x}_{k-1}\right)\right)^{-1} \nabla f\left(\boldsymbol{x}_{k-1}\right)$
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \boldsymbol{l}_k$
  $\Sigma_k \leftarrow \left(\nabla^2 f\left(\boldsymbol{x}_{k-1}\right)\right)^{-1} J_k^T \Sigma_{zz} J_k \left(\nabla^2 f\left(\boldsymbol{x}_{k-1}\right)\right)^{-T}$
until $\|\nabla f\left(\boldsymbol{x}_k\right)\| \leq \epsilon \vee k \geq k_{max}$
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k$
$\Sigma_{\hat{x}\hat{x}} \leftarrow \Sigma_k$

Besides the calculation efforts for the second derivatives, some modifications exist to overcome the drawbacks mentioned above.

Damped Newton Method

Like in the Gradient Descent method, the line-search method can be used to determine the step size ρ_k (see equation (4.104)). However, this method does not prevent ill-conditioned or singular Hessians, so in some cases, it is not possible to determine the direction $\boldsymbol{l}_k$.

Trust Region Newton Method

The line-search method determines, for a given descending direction, a step size that reduces the value of f. *Trust region* methods, however, first select a step size, then determine a direction. In a sense, both methods differ in the order in which they determine the direction and step size.

Trust region methods approximate the objective function f locally in each iteration by a model function m_k, by using the local informations about f at the previous estimate $\boldsymbol{x}_{k-1}$. The new estimate is determined as the minimizer of this model. Because the model function may not be a good approximation for any $\boldsymbol{x}$ far away from $\boldsymbol{x}_{k-1}$, trust region methods restrict the possible values of the new estimate to a region Δ_k, the *trust region*, around the previous estimate $\boldsymbol{x}_{k-1}$.

$$l_k = \arg \min_{\|l\| \leq \Delta_k} m_k\left(l\right) \tag{4.111}$$

If the new cost value $f\left(\boldsymbol{x}_{k-1}+l_k\right)$ is greater than the previous value $f\left(\boldsymbol{x}_{k-1}\right)$, the step is rejected, the trust region radius Δ_k is scaled down and a new estimate is determined. If, however, the new estimate could reduce the cost value, it is accepted and the trust region radius Δ_k is increased for the next iteration.

4.2.3 Gauss-Newton

The *Gauss-Newton* method is a very efficient method that uses only the first derivatives of f. In special cases, it provides quadratic convergence as the Newton method.

The Gauss-Newton method first linearizes $\boldsymbol{r}$, then determines the adjustment vector by setting the derivative of the new cost function f to zero.

$$\boldsymbol{r}\left(\boldsymbol{x}_{k-1}+\boldsymbol{l}_k\right) \approx \boldsymbol{r}_k + J_k^T \boldsymbol{l}_k \tag{4.112}$$

the cost function then becomes:

$$\begin{aligned} f\left(\boldsymbol{x}_{k-1}+\boldsymbol{l}_k\right) &\approx \frac{1}{2}\left(\boldsymbol{r}_k^T\boldsymbol{r}_k + 2\boldsymbol{l}_k^T J_k^T \boldsymbol{r}_k + \boldsymbol{l}_k^T J_k^T J_k \boldsymbol{l}_k\right) \\ &\approx f_k + \boldsymbol{l}_k^T J_k^T \boldsymbol{r}_k + \frac{1}{2}\boldsymbol{l}_k^T J_k^T J_k \boldsymbol{l}_k \end{aligned} \tag{4.113}$$

The first and second derivatives of the approximated f with respect to $\boldsymbol{l}_k$ are given by:

$$\frac{\partial f}{\partial \boldsymbol{l}_k} = J_k^T \boldsymbol{r}_k + J_k^T J_k \boldsymbol{l}_k \tag{4.114}$$

$$\frac{\partial^2 f}{\left(\partial \boldsymbol{l}_k\right)^2} = J_k^T J_k \tag{4.115}$$

Since the second derivative is symmetric, the unique minimizer can be determined by the normal equation if J_k has full rank. The normal equation is given by:

$$J_k^T J_k \boldsymbol{l}_k = -J_k^T \boldsymbol{r}_k \tag{4.116}$$

The unique minimizer is obtained by:

$$\boldsymbol{l}_k = -\left(J_k^T J_k\right)^{-1} J_k^T \boldsymbol{r}_k \tag{4.117}$$

It can be seen that the Hessian is approximated by neglecting the higher order terms in equa-

tion (4.91). Thus, the Gauss-Newton method is an approximation of the Newton method and coincides with it if $\boldsymbol{r}(\boldsymbol{x}^*) = \mathbf{0}$.
If $|r_i(\boldsymbol{x}^*)|$ are small, we can expect superlinear convergence, but in general we will have just linear convergence.
The covariance matrix of the final estimate is given by:

$$\Sigma_{\hat{x}\hat{x}} = \left(J_k^T J_k\right)^{-1} J_k^T \Sigma_{zz} J_k \left(J_k^T J_k\right)^{-1} \tag{4.118}$$

If the covariance matrix of the measurement vector is homoscedastic (see section 4.1.2), the covariance matrix of the estimate reduces to:

$$\begin{aligned}\Sigma_{\hat{x}\hat{x}} &= \left(J_k^T J_k\right)^{-1} J_k^T \sigma_z^2 I_m J_k \left(J_k^T J_k\right)^{-1} \\ &= \sigma_z^2 \left(J_k^T J_k\right)^{-1} J_k^T J_k \left(J_k^T J_k\right)^{-1} \\ &= \sigma_z^2 \left(J_k^T J_k\right)^{-1}\end{aligned} \tag{4.119}$$

However, for the general case $r_i(\boldsymbol{x}^*) \neq 0$ follows:

$$\begin{aligned}\nabla f(\boldsymbol{x}^*) &= 0 \\ J^T(\boldsymbol{x}^*)\,\boldsymbol{r}(\boldsymbol{x}^*) &= 0\end{aligned}$$

Thus, in the vicinity of $\boldsymbol{x}^*$, $J_k^T J_k$ is singular as J_k does not have full rank, and no descending direction can be determined. A solution similar to the Trust Region Newton method is the *Levenberg-Marquardt* method, which is introduced in section 4.2.4. The algorithm for the classical Gauss Newton method is given by Algorithm 4.3.

Algorithm 4.3 Classical Gauss Newton Method

Require: $\boldsymbol{x}_0$, $\epsilon > 0$, $k_{max} > 0$
1: $k \leftarrow 0$
2: **repeat**
3: $\quad k \leftarrow k + 1$
4: $\quad \boldsymbol{l}_k \leftarrow -\left(J_k^T J_k\right)^{-1} J_k^T \boldsymbol{r}_k$
5: $\quad \boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \boldsymbol{l}_k$
6: $\quad \Sigma_k \leftarrow \sigma_z^2 \left(J_k^T J_k\right)^{-1}$
7: **until** $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon \vee k \geq k_{max}$
8: $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k$
9: $\Sigma_{\hat{x}\hat{x}} \leftarrow \Sigma_k$

4.2.3.1 Weighted Gauss-Newton

The above estimate (4.117) is the OLS estimate of the following equation:

$$J_k \boldsymbol{l}_k = -\boldsymbol{r}_k \tag{4.120}$$

It is the OLS estimate of the linearized observation model (4.1).

$$h\left(\boldsymbol{x}_{k-1} + \boldsymbol{l}_k\right) = \boldsymbol{z}$$

Truncated Taylor expansion leads to:

$$\begin{aligned} h\left(\boldsymbol{x}_{k-1}\right) + J_k \boldsymbol{l}_k &= \boldsymbol{z} \\ J_k \boldsymbol{l}_k &= \boldsymbol{z} - h\left(\boldsymbol{x}_{k-1}\right) = -\boldsymbol{r}_k \end{aligned}$$

If the covariance matrix Σ_{zz} of the observation vector $\boldsymbol{z}$ is given and does not satisfy (4.23), the OLS estimate is not BLUE and the GLS solution is more favorable.
The weighted normal equation becomes:

$$\left(J_k^T \Sigma_{zz}^{-1} J_k\right) \boldsymbol{l}_k = -J_k^T \Sigma_{zz}^{-1} \boldsymbol{r}_k \tag{4.121}$$

with the correction step:

$$\boldsymbol{l}_k = -\left(J_k^T \Sigma_{zz}^{-1} J_k\right)^{-1} J_k^T \Sigma_{zz}^{-1} \boldsymbol{r}_k \tag{4.122}$$

This estimator is the nonlinear counterpart to the Gauss-Markov estimator described in section 4.1.2. The covariance matrix of the final estimate is given by:

$$\Sigma_{\hat{x}\hat{x}} = \left(J_k^T \Sigma_{zz}^{-1} J_k\right)^{-1} \tag{4.123}$$

The solution is equivalent to the OLS estimate of the following cost function:

$$\begin{aligned} f\left(\boldsymbol{x}\right) &= \frac{1}{2} \left(h(\boldsymbol{x}) - \boldsymbol{z}\right)^T \Sigma_{zz}^{-1} \left(h(\boldsymbol{x}) - \boldsymbol{z}\right) \\ &= \frac{1}{2} \boldsymbol{r}^T \left(\boldsymbol{x}\right) \Sigma_{zz}^{-1} \boldsymbol{r} \left(\boldsymbol{x}\right) \end{aligned} \tag{4.124}$$

If additional *a priori* knowledge $\tilde{\boldsymbol{x}}$ and $\Sigma_{\tilde{x}\tilde{x}}$ is available, the nonlinear counterpart to the Minimum Variance estimator (see section 4.1.3) can be used in each iteration.

Equation (4.120) is extended to:

$$\begin{pmatrix} J_k \\ I_n \end{pmatrix} \boldsymbol{l}_k = \begin{pmatrix} -\boldsymbol{r}_k \\ \boldsymbol{d}_k \end{pmatrix} \tag{4.125}$$

with the desired change $\boldsymbol{d}_k = \tilde{\boldsymbol{x}} - \boldsymbol{x}_{k-1}$.
And the normal Equation becomes:

$$\left(J_k^T \Sigma_{zz}^{-1} J_k + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right) \boldsymbol{l}_k = -J_k^T \Sigma_{zz}^{-1} \boldsymbol{r}_k + \Sigma_{\tilde{x}\tilde{x}}^{-1} \boldsymbol{d} \tag{4.126}$$

with the covariance matrix of the final parameters

$$\Sigma_{\hat{x}\hat{x}} = \left(J_k^T \Sigma_{zz}^{-1} J_k + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right)^{-1} \tag{4.127}$$

The algorithm for the weighted Gauss-Newton method with prior knowledge is depicted in algorithm 4.4.

Algorithm 4.4 Weighted Gauss Newton Method with Prior Knowledge

Require: $\tilde{\boldsymbol{x}}$, $\Sigma_{\tilde{x}\tilde{x}}$, Σ_{zz} , $\epsilon > 0$, $k_{max} > 0$

$k \leftarrow 0$
$\boldsymbol{x}_0 \leftarrow \tilde{\boldsymbol{x}}$
repeat
  $k \leftarrow k + 1$
  $\boldsymbol{d}_k \leftarrow \tilde{\boldsymbol{x}} - \boldsymbol{x}_{k-1}$
  $\boldsymbol{l}_k \leftarrow \left(J_k^T \Sigma_{zz}^{-1} J_k + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right)^{-1} \left(-J_k^T \Sigma_{zz}^{-1} \boldsymbol{r}_k + \Sigma_{\tilde{x}\tilde{x}}^{-1} \boldsymbol{d}_k\right)$
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \boldsymbol{l}_k$
  $\Sigma_k \leftarrow \sigma_z^2 \left(J_k^T \Sigma_{zz}^{-1} J_k + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right)^{-1}$
until $\| \nabla f(\boldsymbol{x}_k) \| \leq \epsilon \vee k \geq k_{max}$
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k$
$\Sigma_{\hat{x}\hat{x}} \leftarrow \Sigma_k$

4.2.4 Levenberg-Marquardt

As mentioned above, the *Levenberg-Marquardt* method is similar to the Trust Region Newton method and is more robust than the Gauss-Newton method [Mar63]. In fact, the Levenberg-Marquardt method is the progenitor of the Trust Region method for general optimization [NW99].

The restriction of the step size is not explicitly formulated as in equation (4.111), but it is

implicitly contained in the scalar λ_k, which is added to the diagonal elements of the Hessian. The magnitude of λ_k is selected in each iteration to enforce positive definiteness of the so modified Hessian. The augmented normal equation then becomes:

$$\left(J_k^T J_k + \lambda_k I_n\right) \boldsymbol{l}_k = -J_k^T \boldsymbol{r}_k \tag{4.128}$$

In each iteration this scalar varies, so that the Hessian becomes positive definite and a unique correction step $\boldsymbol{l}_k$ can be determined. A typical initial value for λ is $\lambda_1 = 10^{-3}$ [HZ03]. If the current iteration satisfies some descending condition (e.g. equation (4.83)), the correction vector $\boldsymbol{l}_k$ is accepted and λ_k is divided by 10 for the next iteration. Otherwise, if the correction vector increases the error, it is rejected, λ_k is multiplied by 10, and the equation is solved again for the new λ_k. This process is repeated until a suitable λ_k is found that decreases the error, or λ_k reaches an upper bound λ_{max}. The reason of varying λ_k in each iteration is to enforce convergence for this method, while getting a high convergence rate.

This method switches smoothly between the Gauss-Newton method and the Gradient-Descent method, and benefits from the advantages of both: large radius of convergence and high convergence rate.

If λ_k is close to zero the method coincides with the Gauss-Newton method and has quadratic or superlinear convergence. On the other hand, if λ_k is sufficiently large, the diagonal elements of the Hessian dominate the Hessian, and the direction of the correction vector is close to the Gradient Descent method, which has a larger radius of convergence.

The covariance matrix of the estimated parameters are given by equation (4.119), as in practice, it makes no sense to consider the stabilization terms λ_k.

A different implementation is applied by Lowe et al. [Low91]. Given a weight matrix W and a desired correction vector $\boldsymbol{d}$, equation (4.120) is extended to:

$$\begin{pmatrix} J_k \\ \mu W \end{pmatrix} \boldsymbol{l}_k = \begin{pmatrix} -\boldsymbol{r}_k \\ \boldsymbol{d} \end{pmatrix} \tag{4.129}$$

The normal equation becomes:

$$\left(J_k^T J_k + \mu^2 W^T W\right) \boldsymbol{l}_k = -J_k^T \boldsymbol{r}_k + \mu W^T \boldsymbol{d} \tag{4.130}$$

Lowe proposes a diagonal matrix for W with $W_{ii} = \frac{1}{\sigma_{ii}}$. σ_{ii} is the standard deviation of parameter vector x_i, which is determined from the covariance matrix of the previous iteration.

Algorithm 4.5 Levenberg-Marquardt Method

Require: $\boldsymbol{x}_0$, $\epsilon > 0$, $k_{max} > 0$, $\lambda_{max} \geq 10^6$

$k \leftarrow 0$
$\lambda_1 = 10^{-3}$
$sse = f(\boldsymbol{x}_0)$
repeat
  $k \leftarrow k + 1$
  $\boldsymbol{l}_k \leftarrow -\left(J_k^T J_k + \lambda_k I_n\right)^{-1} J_k^T \boldsymbol{r}_k$
  $\boldsymbol{x}_k \leftarrow \boldsymbol{x}_{k-1} + \boldsymbol{l}_k$
  if $f(\boldsymbol{x}_k) > sse$ **then**
    $\lambda_k \leftarrow \lambda_k \cdot 10$
    $k \leftarrow k - 1$
  else
    $sse \leftarrow f(\boldsymbol{x}_k)$
    $\lambda_{k+1} \leftarrow \lambda_k \div 10$
    $\Sigma_k \leftarrow \sigma_z^2 \left(J_k^T J_k\right)^{-1}$
  end if
until $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon \vee k \geq k_{max} \vee \lambda_k \geq \lambda_{max}$
$\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}_k$
$\Sigma_{\hat{x}\hat{x}} \leftarrow \Sigma_k$

4.2.4.1 Weighted Levenberg Marquardt

If the covariance matrix Σ_{zz} of the observation vector $\boldsymbol{z}$ is given and does not satisfy equation (4.23), the normal equation can be replaced by the weighted normal equation (4.121), while the rest of the algorithm stays unchanged, except for the covariance matrix, which is given by equation (4.123), and the evaluation of the new cost value, which is given by equation (4.124).

4.2.5 Outlier Treatment

Least squares estimation methods are very sensitive to outliers. Outliers are defined to be measurements that do not fit to the observation model.

Methods that are capable of determining reliable estimates in the presence of outliers in the measurements are called robust. Robust methods are often characterized by a measure called the *breakdown point* [Hub81]. The breakdown point is defined as the smallest percentage of outliers that can cause the estimate to an arbitrarily large error. Least squares methods, for example, are very sensitive to outliers and even one outlier can lead to considerable errors in the estimates. The breakdown point for least squares estimators is $\frac{1}{m}$, m being the number of observations, which leads to an asymptotic breakdown point of 0%.

Another measure for robustness is the so-called *influence function*, which reflects the change of an estimate caused by inserting an outlier as a function of the distance of the data from the *outlier-free* estimate. The influence function of the least squares estimate is simply proportional to the distance of the point from the estimate (see section M-Estimator).

4.2.5.1 RANSAC

A very effective method for robust estimation, with the highest possible breakdown point of 50%, is the RAndom SAmple Consensus (*RANSAC*) introduced by Fischler et al. [FB81], which is widely used in the field of computer vision. The key idea is simple but powerful.

First, an estimate is determined from a minimal subset of randomly selected measurements, which is called the *minimal solution*. Then the quality of the minimal solution is determined from the fit of all measurements to the observation model using a utility function. Usually, the number of inliers is used, where an inlier is defined to be a measurement whose residual does not exceed a given threshold. This process is repeated until the likelihood of finding a better subset becomes low, and the estimate leading to the best fit is returned. Usually, the estimated parameters are used as initial values for a further least squares refinement using all inliers.

In this thesis, we will use RANSAC for robust *Homography* estimation [HZ03].

4.2.5.2 LMS

Another very robust method is the *least median of squares* (LMS) estimator [Rou84, Rou87], as the median is insensible to outlier (breakdown point of 50%). As the name indicates, LMS determines the estimate which delivers the smallest median of the squared residuals over all observations. This can be done only in a combinatorial way which usually leads to high computational costs [EHPM04]. However, the convergence speed can be increased by several methods. One is to carry out random samples of measurement subsets in a way similar to RANSAC. In this case, the main difference to RANSAC is the rating of an estimate. The advantage of the LMS method compared to RANSAC is that the LMS method does not require *a priori* knowledge about the measurement variance (threshold values)[TM97].

4.2.5.3 Hough Transform

A very popular method in computer vision is the *Hough Transformation* [Hou62]. The parameter space is discretized and represented by an accumulator field. For each measurement all possible parameters are determined that confirm the current measurement. The entries of the accumulator field where the determined parameters fall in are then incremented. The optimal

estimate is expected to be one of the peaks (local maxima) in the accumulator field. Usually, the peaks are found via clustering techniques [Zha95]. The breakdown point of the Hough transformation depends on the discretization level, and can almost reach 50% [GZ05].

Nevertheless, the determination of all parameters for each measurement, confirming the current measurement, is not trivial for complex or high-dimensional models. And even the straightforward method of iterating through all entries of the accumulator field is insufficient for high-dimensional models. Another drawback is that this method requires discretization in the parameter space, which usually leads to limited accuracy.

However, this method is very efficient for low-dimensional parameter spaces and simple models. In the context of this thesis, Hough transformation is used for robust line detection (see section 5.5.1.1)..

4.2.5.4 M-Estimators

The main method used in this thesis for outlier treatment is the class of the so-called *maximum likelihood type estimators* or *M-Estimators* [Hub81], as it can be integrated into a least squares estimation. In general, M-Estimator have a breakdown point of 0%. In practical applications, however, where a good initial guess is given or the outlier rate is moderate, M-Estimators provide a good trade-off between robustness and efficiency.

As the residuals of most outliers are large compared to the residuals of correct measurements, M-Estimator suppress measurements with large residuals.
The cost function (4.80)

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{m} r_i^2$$

is replaced by another function of the residuals to limit the impact of outliers.

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} \varphi(r_i) \tag{4.131}$$

where φ is a symmetric positive definite function with an unique minimum at 0 and is less increasing than square. The minimum of the new cost function is determined by setting its first derivative to 0.

$$\sum_{i=1}^{m} \psi(r_i)\frac{\partial r_i}{\partial x_j} = 0 \quad \text{for} \quad j = 1, \ldots, n \tag{4.132}$$

The derivative $\psi(r) = \frac{\partial \varphi(r)}{\partial r}$ is called *influence function*.
By defining the *weight function* as

$$w(r) = \frac{\psi(r)}{r} \tag{4.133}$$

above equation (4.132) becomes:

$$\sum_{i=1}^{m} w\left(r_i^{(k-1)}\right) r_i \frac{\partial r_i}{\partial x_j} = 0 \quad \text{for} \quad j = 1, \ldots, n \tag{4.134}$$

This can be formulated as an iterated reweighted least squares problem with following cost function:

$$f(\boldsymbol{x}) = \frac{1}{2} \sum_{i=1}^{m} w\left(r_i^{(k-1)}\right) r_i^2 \tag{4.135}$$

The weights $w\left(r_i^{(k-1)}\right)$ are calculated from the residuals of the previous iteration $k-1$.
Writing this in matrix form with the diagonal weight matrix $W\left(W_{ii} = w\left(r_i^{(k-1)}\right)\right)$, the cost function can be written as:

$$f(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{r}^T W \boldsymbol{r} \tag{4.136}$$

and the estimate can be found using one of the nonlinear weighted least squares method described in previous sections.

Weight Functions

A variety of weight functions are designed and tested for different purposes [Zha95]. In the computer vision community, the M-Estimators by Tukey [FW77] and Huber [Hub81] are used commonly and will be introduced in this section. Further on, we will present a new weight function, that changes its shape in each iteration.
Tukey uses the following function:

$$\varphi(x) = \begin{cases} \frac{a^2}{6}\left(1 - \left[1 - \left(\frac{x}{a}\right)^2\right]^3\right) & \text{for} \quad |x| \leq a \\ \frac{a^2}{6} & \text{for} \quad |x| > a \end{cases} \tag{4.137}$$

with corresponding weight function:

$$w(x) = \begin{cases} \left[1 - \left(\frac{x}{a}\right)^2\right]^2 & \text{for} \quad |x| \leq a \\ 0 & \text{for} \quad |x| > a \end{cases} \tag{4.138}$$

Measurements whose residuals exceed the threshold value a are completely eliminated. A less restrictive function is proposed by Huber [Hub81].

$$\varphi(x) = \begin{cases} \frac{x^2}{2} & \text{for} \quad |x| \leq a \\ a\left(|x| - \frac{a}{2}\right) & \text{for} \quad |x| > a \end{cases} \tag{4.139}$$

The corresponding weight function is given by:

$$w(x) = \begin{cases} 1 & \text{for} \quad |x| \leq a \\ \frac{a}{|x|} & \text{for} \quad |x| > a \end{cases} \tag{4.140}$$

Huber's weight function does not eliminate the influence of outliers completely, but reduces their impact from a quadratic influence to a linear one. However, the disadvantage of this weight function is that the final estimate is still corrupted by the outliers, even if their influence is limited.

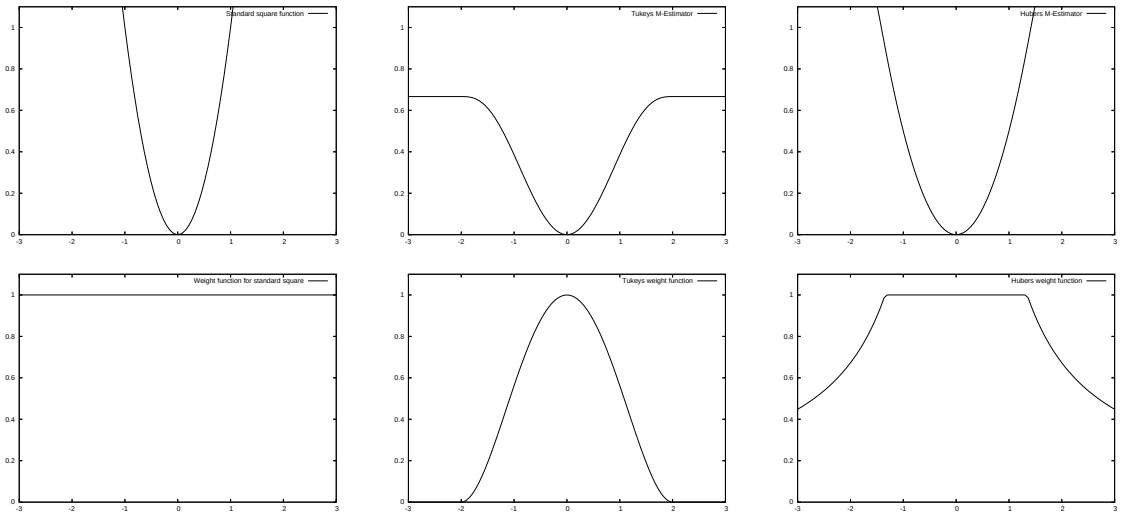


Figure 4.2 Commonly used weight functions: $\varphi(x)$ and weight function $w(x)$ for least squares (left), Tukey (center) and Huber (right)

Proposed weight function

We propose a weight function which has been used successfully in the update stage of the *Incremental Camera Calibration* module. It has been shown to generate very reliable results in our experimental evaluations.
The function is specified by the parameters a and b.

$$w(x) = \frac{1}{\left(\frac{|x|}{a}\right)^b + 1} \tag{4.141}$$

Parameter a specifies the value of x where the weight function becomes 0.5 and is comparable to the parameters of above weight functions, whereas parameter b indicates the gradient of the curve at $x = a$. For $b = 2$, above weight function is equivalent with Cauchy's weight function (see [Zha95]).

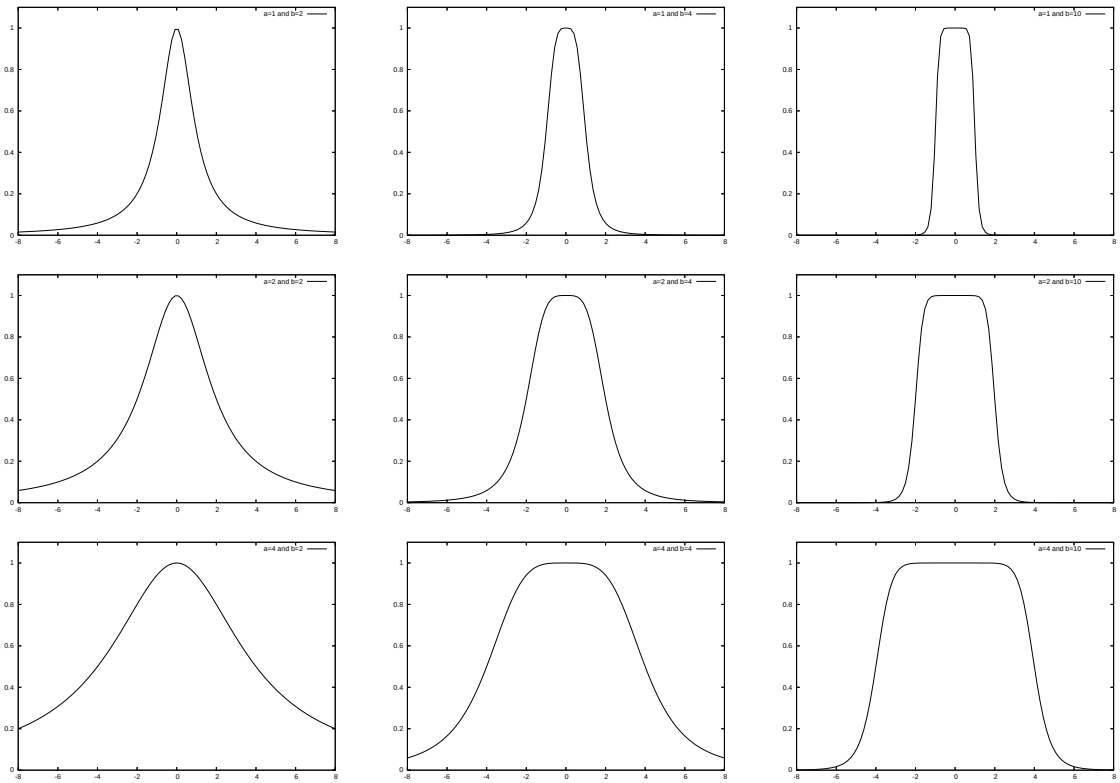


Figure 4.3 Proposed weight function for different parameters a and b: from left to right $b = 2$, $b = 4$, $b = 10$; from top to bottom $a = 1$, $a = 2$, $a = 4$

The main idea is to select a small b at the beginning and to increase it in each iteration. In the first iterations the weight function is less restrictive and is comparable to Huber's function. I.e. measurements with large residuals are not eliminated completely. Since the state vector is

estimated more and more accurately in each iteration, the weight function gets more and more restrictive. In the final iterations, the influence of outliers is almost completely eliminated, while all inliers are weighted almost equally. The initial value and the increment of b can be selected dependent on the application. In the update stage of the camera tracking task, an initial value of 2 and an increment by 2 provided robust results.

The rows in figure 4.3, for example, show the weight function for the first, second and 5th iteration for different clipping factors a.

Clipping factor a

The above weight functions need to be parameterized with a clipping factor a, which leads to another problem: How should a be selected?

An intuitive way of selecting a is to scale it with the standard deviation σ_r of the residuals. For example, a is selected to be $a = 4.6851 \cdot \sigma_r$ for Tukey's weight function and $a = 1.345 \cdot \sigma_r$ for Huber's weight function [Zha95]. For the proposed weight function, the clipping factor was selected to be $2\sigma_r$, which worked reliably in the context of this thesis.

The scale σ_r must be provided *a priori* from the analysis of the sensor or the process from which the measurements were obtained. In most cases, however, σ_r is unknown and must be estimated from the measurements. Determining σ_r by the empirical variance is infeasible, since outlier have a strong impact on $\hat{\sigma}_r$. Thus, the estimate of σ_r must be robust as well.

Robust Estimation of σ_r

A robust estimate for σ_r is given by

$$\hat{\sigma}_r = 1.4826 \cdot \text{median}\left(|r_i|\right) \tag{4.142}$$

Intuitively, this can be explained as follows: Since the residuals r_i are assumed to be normally distributed, $\frac{r_i^2}{\sigma_r^2}$ are supposed to be χ_1^2 distributed with one degree of freedom. Like the 50% quantile of the χ_1^2 distribution, the median separates the residuals into two parts with equal size. And since the median is very robust against outliers, this equality is used to obtain a

robust estimation of σ_r.

$$\begin{aligned}
\text{median}\left(\frac{r_i^2}{\sigma_r^2}\right) &= \chi^2_{1;0.5} \\
\frac{\text{median}\left(r_i^2\right)}{\sigma_r^2} &= 0.454936 \\
\sigma_r^2 &= \frac{\text{median}\left(r_i^2\right)}{0.454936} \\
\hat{\sigma}_r &= \frac{\text{median}\left(|r_i|\right)}{0.67449} = 1.4826 \cdot \text{median}\left(|r_i|\right)
\end{aligned} \tag{4.143}$$

The following robust estimate compensates for the effects caused by small data sets [Rou87]:

$$\hat{\sigma}_r = 1.4826 \cdot \left(1 + \frac{5}{m-n}\right) \cdot \text{median}\left(|r_i|\right) \tag{4.144}$$

4.3 Maximum Likelihood Estimator

The above methods define the *optimal* estimate as the one which minimizes the variance of the estimate. The *Maximum Likelihood* or *ML* estimator defines the optimal estimate as the one which maximizes the likelihood of the observations.

Let $p\left(\boldsymbol{z}|\boldsymbol{x}\right)$ denote the likelihood of the observations, then the ML estimate is given by:

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}} p\left(\boldsymbol{z}|\boldsymbol{x}\right) \tag{4.145}$$

Assuming Gaussian measurement errors in $\boldsymbol{z}$ with covariance matrix Σ_{zz} the likelihood function is given by:

$$p\left(\boldsymbol{z}|\boldsymbol{x}\right) = \frac{1}{\sqrt{|2\pi\Sigma_{zz}|}} e^{-\frac{1}{2}(H\boldsymbol{x}-\boldsymbol{z})^T \Sigma_{zz}^{-1} (H\boldsymbol{x}-\boldsymbol{z})} \tag{4.146}$$

Since the natural logarithm is a strictly monotonic increasing function, the maximizer of the likelihood function is identical to the maximizer of the *log likelihood* and therefore identical to the minimizer of the negative *log likelihood*.

$$\begin{aligned}
\hat{\boldsymbol{x}} &= \arg\max_{\boldsymbol{x}} -\frac{1}{2}\left(H\boldsymbol{x}-\boldsymbol{z}\right)^T \Sigma_{zz}^{-1}\left(H\boldsymbol{x}-\boldsymbol{z}\right) - \frac{1}{2}\ln|2\pi\Sigma_{zz}| \\
&= \arg\min_{\boldsymbol{x}} \frac{1}{2}\left(H\boldsymbol{x}-\boldsymbol{z}\right)^T \Sigma_{zz}^{-1}\left(H\boldsymbol{x}-\boldsymbol{z}\right) + \frac{1}{2}\ln|2\pi\Sigma_{zz}|
\end{aligned}$$

The factor $\frac{1}{2}$ and the term $\ln\left(|2\pi\Sigma_{zz}|\right)$ are independent of x, thus have no influence on the

minimizer[3], and can be removed.

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \left(H\boldsymbol{x} - \boldsymbol{z}\right)^T \Sigma_{zz}^{-1} \left(H\boldsymbol{x} - \boldsymbol{z}\right)$$

The resulting estimator minimizes the Mahalanobis distance of the residual vector and is the same problem as described in section 4.1.2. Thus the *Maximum Likelihood* estimate for Gaussian measurement errors is equivalent to the *Gauss-Markov* estimate for arbitrary distributions.

4.4 Maximum A Posteriori Estimator

If *a priori* information about the system state in form of a pdf $p\left(\boldsymbol{x}\right)$ is available, the posterior distribution of $\boldsymbol{x}$ can be determined by applying *Bayes' Rule*.

$$p\left(\boldsymbol{x}|\boldsymbol{z}\right) = \frac{p\left(\boldsymbol{z}|\boldsymbol{x}\right) p\left(\boldsymbol{x}\right)}{p\left(\boldsymbol{z}\right)} \tag{4.147}$$

The *Maximum A Posteriori* or *MAP* estimation estimates the mode of this posterior distribution. As the denominator on the right hand side is independent of $\boldsymbol{x}$, it can be removed from the equation and the MAP estimation is given by

$$\hat{\boldsymbol{x}} = \arg\max_{x} p\left(\boldsymbol{z}|\boldsymbol{x}\right) p\left(\boldsymbol{x}\right) \tag{4.148}$$

If the *a priori* density $p\left(\boldsymbol{x}\right)$ is assumed to be uniform, thus infinitesimally small but constant, the above estimate is independent of $p\left(\boldsymbol{x}\right)$ and the estimator coincides with the Maximum Likelihood estimator (see section 4.3).
If all quantities are assumed to be Gaussian, the equation can be explicitly written as:

$$\hat{\boldsymbol{x}} = \arg\max_{x} \frac{1}{\sqrt{|2\pi\Sigma_{zz}|}} e^{-\frac{1}{2}(H\boldsymbol{x}-\boldsymbol{z})^T \Sigma_{zz}^{-1} (H\boldsymbol{x}-\boldsymbol{z})} \frac{1}{\sqrt{|2\pi\Sigma_{\tilde{x}\tilde{x}}|}} e^{-\frac{1}{2}\left(\boldsymbol{x}-\tilde{\boldsymbol{x}}\right)^T \Sigma_{\tilde{x}\tilde{x}}^{-1} \left(\boldsymbol{x}-\tilde{\boldsymbol{x}}\right)} \tag{4.149}$$

Using the negative log likelihood and removing constants leads to the following estimator:

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) \tag{4.150}$$

with

$$f\left(\boldsymbol{x}\right) = \frac{1}{2}\left(H\boldsymbol{x} - \boldsymbol{z}\right)^T \Sigma_{zz}^{-1} \left(H\boldsymbol{x} - \boldsymbol{z}\right) + \frac{1}{2}\left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right)^T \Sigma_{\tilde{x}\tilde{x}}^{-1} \left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right)$$

[3] but they influence to the minimum

Another advantage of using the negative log likelihood is that the product of probabilities, which can get close to zero, becomes a sum and therefore is numerically more stable.

The necessary conditions for a minimum of the above term are given by equations (4.10) and (4.11).

$$\begin{aligned}\frac{\partial f}{\partial \boldsymbol{x}} &= H^T \Sigma_{zz}^{-1} \left(H\boldsymbol{x} - \boldsymbol{z}\right) + \Sigma_{\tilde{x}\tilde{x}}^{-1} \left(\boldsymbol{x} - \tilde{\boldsymbol{x}}\right) = 0 \\ &= \left(H^T \Sigma_{zz}^{-1} H + \Sigma_{\tilde{x}\tilde{x}}^{-1}\right) \boldsymbol{x} + H^T \Sigma_{zz}^{-1} \boldsymbol{z} + \Sigma_{\tilde{x}\tilde{x}}^{-1} \tilde{\boldsymbol{x}} = 0 \qquad (4.151)\\ \frac{\partial^2 f}{\partial \boldsymbol{x}^2} &= H^T \Sigma_{zz}^{-1} H + \Sigma_{\tilde{x}\tilde{x}}^{-1} > 0 \qquad (4.152)\end{aligned}$$

Assuming that we have positive definite covariance matrices and that H has full rank, the sufficient condition (4.11) is met.

Thus the *MAP* estimate for Gaussian quantities is equivalent to the *Minimum Variance* estimator for arbitrary distributions as described in section 4.1.3. Therefore, the relation between *Gauss-Markov* estimator and *Minimum Variance* estimator also holds between the *Maximum Likelihood* estimator and the *MAP* estimator: The *MAP* estimate coincides with the *Maximum Likelihood* estimate, if the prior is *non-informative* (i.e. $\Sigma_{\tilde{x}\tilde{x}} \to \infty$).

4.5 Bayesian Filters for Gaussian Quantities

In section 4.4 we have used *Bayes' Theorem* to derive the MAP estimator. *Bayesian Filters* are a general class of recursive estimators based on Bayes' Theorem that will be introduced in this section. The main idea is to estimate the posterior probability density over the state space, given all measurement data [Ast65].

Let the posterior pdf of current estimate $\hat{\boldsymbol{x}}_t$ be denoted by

$$\begin{aligned}Bel(\hat{\boldsymbol{x}}_t) &= p\left(\hat{\boldsymbol{x}}_t | \boldsymbol{z}_{1:t}, \boldsymbol{u}_{1:t}\right) \\ &= p\left(\hat{\boldsymbol{x}}_t | \boldsymbol{z}_t, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) \qquad (4.153)\end{aligned}$$

By applying Bayes' Theorem, we obtain:

$$Bel(\hat{\boldsymbol{x}}_t) = \frac{p\left(\boldsymbol{z}_t | \hat{\boldsymbol{x}}_t, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) p\left(\hat{\boldsymbol{x}}_t | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)}{p\left(\boldsymbol{z}_t | \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)} \qquad (4.154)$$

Using the Markov or Complete World assumption (4.1), we get:

$$Bel(\hat{\boldsymbol{x}}_t) = \frac{p\left(\boldsymbol{z}_t|\hat{\boldsymbol{x}}_t\right) p\left(\hat{\boldsymbol{x}}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)}{p\left(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)} \tag{4.155}$$

The pdf $p\left(\hat{\boldsymbol{x}}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)$ on the right hand side is the predictor for the current state and is called the *time evolution* or *time update* stage. As no new measurements are available for this step, this must be done with the system or motion model (4.2), which is denoted by:

$$p\left(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t\right) \tag{4.156}$$

and we get:

$$p\left(\hat{\boldsymbol{x}}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) = \int p\left(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) p\left(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) d\hat{\boldsymbol{x}}_{t-1} \tag{4.157}$$

which is known as the *Chapman-Kolmogorov* equation. Intuitively, it can be understood as the sum of the probabilities of all paths from the last estimate to the current estimate over the motion model.

As the Markov assumption also states that the best predictor of the current state is given by the previous estimate $\hat{\boldsymbol{x}}_{t-1}$ and current control $\boldsymbol{u}_t$, we get the final recursive form:

$$\begin{aligned} Bel\left(\hat{\boldsymbol{x}}_t\right) &= \frac{p\left(\boldsymbol{z}_t|\hat{\boldsymbol{x}}_t\right)}{p\left(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right)} \int p\left(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t\right) p\left(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) d\hat{\boldsymbol{x}}_{t-1} \\ &= \eta p\left(\boldsymbol{z}_t|\hat{\boldsymbol{x}}_t\right) \int p\left(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t\right) p\left(\hat{\boldsymbol{x}}_{t-1}|\boldsymbol{z}_{1:t-1}, \boldsymbol{u}_{1:t}\right) d\hat{\boldsymbol{x}}_{t-1} \\ &= \eta p\left(\boldsymbol{z}_t|\hat{\boldsymbol{x}}_t\right) \int p\left(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t\right) Bel\left(\hat{\boldsymbol{x}}_{t-1}\right) d\hat{\boldsymbol{x}}_{t-1} \end{aligned} \tag{4.158}$$

with η being a normalizing constant.

Given the recursive character of the Bayes Filter, it is obvious that a prior distribution $Bel\left(\hat{\boldsymbol{x}}_0\right)$ is required. If no such initial guess is available, one can provide an arbitrary estimate with a very large uncertainty (covariance matrix). In this case, however, the first reliable estimate can be determined after n independent measurements, n being the dimension of the state vector.

4.5.1 Linear Kalman Filter

Gaussian Bayes Filters assume that all quantities are normally distributed and can be represented by their mean and covariance matrix. The *Kalman Filter* belongs to the class of

Gaussian Bayes Filters.

$$Bel(\hat{\boldsymbol{x}}_t) = N\left(\hat{\boldsymbol{x}}_t; \bar{\boldsymbol{x}}_t, \Sigma_{\hat{x}_t\hat{x}_t}\right) \tag{4.159}$$

Since the convolution and product of two Gaussians is again a Gaussian, the *time update* and *measurement update* deliver normally distributed random variables [Pri].

Assuming linear motion and observation models, (4.3) and (4.4), the prediction stage is given by equations (4.75) and (4.76), whereas the measurement update stage is given by equations (4.77)-(4.79).

The Linear Kalman Filter algorithm is given by algorithm 4.6.

Algorithm 4.6 Linear Kalman Filter

Require: $< \hat{\boldsymbol{x}}, \hat{P} >$ % prior distribution
loop
  if time update **then**
    $\tilde{\boldsymbol{x}} \leftarrow A\hat{\boldsymbol{x}} + B\boldsymbol{u}$
    $\tilde{P} \leftarrow A\hat{P}A^T + R$
  else if measurement update **then**
    $K \leftarrow \tilde{P}H^T\left(H\tilde{P}H^T + Q\right)^{-1}$
    $\hat{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + K\left(\boldsymbol{z} - H\tilde{\boldsymbol{x}}\right)$
    $\hat{P} \leftarrow \left(I_n - KH\right)\tilde{P}$
  end if
end loop

4.5.2 Extended Kalman Filter

The classical Kalman Filter (LKF) assumes linear motion and observation models. However, in most applications, neither the motion model nor the observation model are linear. For example, in the context of this thesis, where camera parameters have to be estimated by using images taken from a camera, the mapping of measurements from the image to the parameter space is obviously not linear.

To overcome this problem, Jazwinski proposed the *Extended Kalman Filter* (EKF) [Jaz70]. Nonlinear mappings are approximated by the truncated Taylor expansion as described in (2.15) to propagate uncertainties.

The time update stage becomes:

$$\tilde{\boldsymbol{x}}_t = g(\hat{\boldsymbol{x}}_{t-1}, \boldsymbol{u}_t) \tag{4.160}$$

$$\tilde{P}_t = G_t \hat{\boldsymbol{x}}_{t-1} G_t^T + R \tag{4.161}$$

and the measurement update stage becomes:

$$K_t = \tilde{P}_t H_t^T \left(H_t \tilde{P}_t H_t^T + Q_t \right)^{-1} \tag{4.162}$$

$$\hat{\boldsymbol{x}}_t = \tilde{\boldsymbol{x}} + K_t \left(\boldsymbol{z}_t - h(\tilde{\boldsymbol{x}}_t) \right) \tag{4.163}$$

$$\hat{P}_t = \left(I_n - K_t H_t \right) \tilde{P}_t \tag{4.164}$$

with G and H being the Jacobian of the motion model and observation model respectively.

As mentioned in section 2.3.1.2, the quality of the approximation strongly depends on the linearity of the transformation function around the seed value. If the function has high curvature at the seedpoint, the propagation is inaccurate and therefore the estimate can be far from the true state. By using second or higher order terms in the Taylor expansion, more accurate estimates can be achieved. These Kalman Filters are called second order or higher order Kalman Filter (HKF) [BSF88]. Since the computational costs compared to the accuracy increase inadequately with the use of higher order terms, HKFs are used rarely [LBS01].

4.5.3 Iterative Extended Kalman Filter

The EKF gets inaccurate, if the transformation functions have high curvature. This drawback can partly be removed by the *Iterative Extended Kalman Filter*, which makes a better tradeoff between accuracy and efficiency than HKFs. The key idea is the same as for nonlinear least squares methods, described in section 4.2. Unlike the EKF, which linearizes and determines the estimate in the update stage only once, the IEKF repeats the update stage several times. In each iteration, the observation model is linearized around the previous estimate and a new estimate is determined.

As the measurement update stage coincides with a nonlinear MAP estimation [BSF88, Jaz70, Gel74], the updated state vector can be determined by a nonlinear least squares method as described in section 4.2.

In the context of this thesis, the IEKF is used to track the camera parameters during a soccer game. The MAP estimation is determined by an extended version of the weighted Gauss-Newton method to overcome outliers in the measurements.

4.6 Conclusion

In this chapter we provided a survey about commonly used parameter estimation techniques. Particularly, we introduced several linear and nonlinear least squares estimator, and compared them to other approaches. It was shown, that the minimum variance estimator for arbitrary distributions is equivalent to the MAP estimator for gaussian distributions. Hence, the Kalman filter can be derived as the recursive linear least squares estimator [BS89, Fau93, Sor70, Sor80] for arbitrarily distributed errors, or as a gaussian Bayes filter [BSF88, BP99, TBF05] for normally distributed errors. Therefore, if the errors are normally distributed, the Kalman filter gives the MAP estimate, but at least the minimum variance estimate if the errors are not normally distributed.

Furthermore, we presented several robust methods and proposed a new weight function for the so-called M-Estimators, which has shown to work reliable in the context of this thesis.

Chapter 5 Camera Tracking

The main subject of this thesis is the automatic initialization and robust tracking of camera parameters (camera calibration) in soccer video streams from TV broadcasting cameras.

Camera calibration determines the parameters of a camera model that best describe the mapping of a physical camera. As the image formation process maps the 3-D world to a 2-D image, information is lost and the inverse transformation can not be uniquely determined from a single image without *a priori* knowledge. In camera calibration this *a priori* knowledge is usually provided by the model of a mapped object in the image. To obtain the camera parameters, a set of known real world features and corresponding projections in the image have to be identified. However, in TV broadcasts the cameras are not accessible or controlable and the only available data is the video stream captured by the camera of interest, which makes standard calibration methods infeasible.

In the case of soccer, the field provides a set of markings, which are specified by the official soccer rules [FIF07] and provide a good feature to carry out camera calibration. Unfortunately, the field size may vary from stadium to stadium, as it is not completely specified by the official FIFA regulations [FIF07] and must be estimated also.

Estimating all parameters repeatedly for every frame is unreliable, inefficient and unnecessary, as most parameters stay almost constant throughout the whole game. TV cameras are usually attached on tripods around the field at a height of $8m - 20m$ and are fixed in their position, while they are actively rotating and zooming to track the game. The main camera is placed on the center of the main stands, whereas the other two major cameras are shifted along the main stands to cover the goal areas. In some cases, the so-called *Tactical cameras* are used to cover large areas of the field and are placed behind the goals at a large height.

In the context of this thesis, we will consider only the above described camera configurations. This allows to make reasonable assumptions to break down the complex task of camera calibration into smaller and easier manageable subproblems. This assumptions and simplifications are described in the following.

The camera is approximated by a pinhole camera model with radial distortion described by

only one coefficient and has rectangular pixel elements. Furthermore, it is assumed that the only changing parameters during the game are the camera's pan and tilt angles and its focal length, which will be called *dynamic parameters* from now on. All other unknown parameters, called *constant parameters*, are assumed to be constant during the whole game and can be determined once beforehand. These simplifications are reasonable, as the effects caused by them are negligibly small.

For example, it is not guaranteed that the camera center lies on both rotation axes. A rotation would cause a displacement of the camera center, and considering the size of typical TV cameras, this displacement could be up to $70cm$ for a $180°$ rotation. However, as the mapped objects (field and player) have a large distance to the camera compared to its focal length, the parallax effects caused by the displacement are negligibly small. Also the principal point and the radial distortion may change while zooming. However, since the principal point has a small impact to the parameter estimation [KH90, KH94], and the radial distortion of TV cameras are usually very small, we can also neglect these effects.

By using the above assumptions the problem is divided in two stages consisting of interacting modules. In the first or the *pre-game* stage, all constant parameters and color information are determined by a mostly automatic process before the game. These information are provided to the modules in the second stage, where the dynamic parameters are estimated during the game. Due to this breakdown, the tracking task achieves high efficiency as well as high reliability, as its state space is reduced considerably.

The modules and their interactions between each other are depicted in figure 5.1.

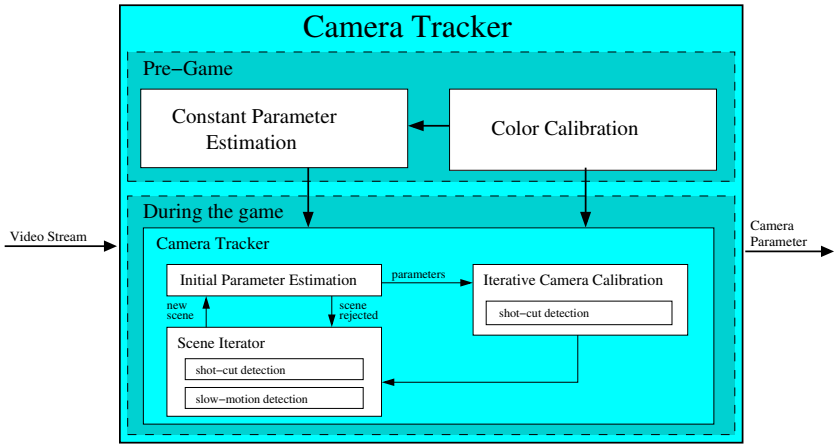


Figure 5.1 The camera tracking system

- The *Color Calibration* module is executed during initialization and therefore belongs to the *pre-game* stage. It determines automatically the color of the field and provides this

information to the other modules.

- The *Constant Parameter Estimation* determines all necessary parameters, which stay constant during the game and is also performed beforehand. The estimated parameters are: the principal point, pixel size, position and roll angle of the camera, and field size.

- The *Scene Iterator* segments the video stream into small sequences or *video segments* at shot boundaries. It iterates successively through all video segments and filters *out of field views*, *close up views* and *slow motion replays*. Segments which are classified as *fieldview*, excluding *slow motion replays*, are passed to the *Initial Parameter Estimator*.

- The *Initial Parameter Estimator* estimates the initial dynamic parameters (pan, tilt, zoom) for each new video segment from extracted image lines. If the image lines are inconsistent with the expected projections of the model lines, the video segment is rejected and the *Scene Iterator* searches for the next suitable sequence. Otherwise, the initial parameters are passed to the *Iterative Camera Calibration*, which successively estimates the camera parameters for the whole sequence.

- The *Iterative Camera Calibration* estimates successively the dynamic parameters until a shot boundary is detected. This is efficiently done by using a modified version of the *Iterated Extended Kalman Filter*. After a shot boundary is detected, this module returns the control to the *Scene Iterator*, which searches for the next suitable sequence.

5.1 Environment Model

The environment model is essential for the camera calibration and tracking tasks, as it describes the only reliable features in real world coordinates.

It describes the soccer field and consists of 2D curves separating regions which differ in color distribution. A field curve is characterized by a two-dimensional curve function describing the shape of the curve, and a color transition describing the color distributions on both sides of the curve.
We distinguish between the following three types of field curves.

- *Field lines* represent the white markings which are completely specified by the official FIFA regulations [FIF07] and therefore are the most reliable features. They are described by two parallel edges at a distance of $12cm$ (line width) and separate the line color between both edges from the field color.

- *Field edges* describe the outer boundary of the field and separate the field from the banner; thus they usually separates field color from non-field color.

- *Lawn lines* do not belong to the official regulations, but are used in most games as supporting lines for the referees. They are purposely caused by the lawn mower, have usually a distance of $3 - 6m$ to each other and separate brighter regions from darker ones.

The necessary features used by all modules are the field lines, whereas the latter two features are optional and are used only by the *Iterative Camera Calibration* module if specified by the user. As the visibility of *Lawn Lines* depends on the perspective, each camera is associated to an own field model.

The field model with its known metrics is depicted in figure 5.3. Projections from different perspectives of the field model are shown in figure 5.2, where *Field edges* are displayed as blue lines, *Lawn lines* as green lines and *Field lines* are depicted as black lines. Red lines show features which are important for the Constant Parameter Estimation module, but are not used during the tracking task.

The world coordinate system is right-handed and defined as follows.

- The origin coincides with the center mark of the field.

- The z-axis is parallel to the normal of the play ground, and is faced upwards.

- The y-axis is parallel to the center line (or halfway line), and is directed away from the main camera.

Straight lines, which are parallel to the y-axis are referred as *vertical* model lines, whereas straight lines which are parallel to the x-axis are called *horizontal* model lines.

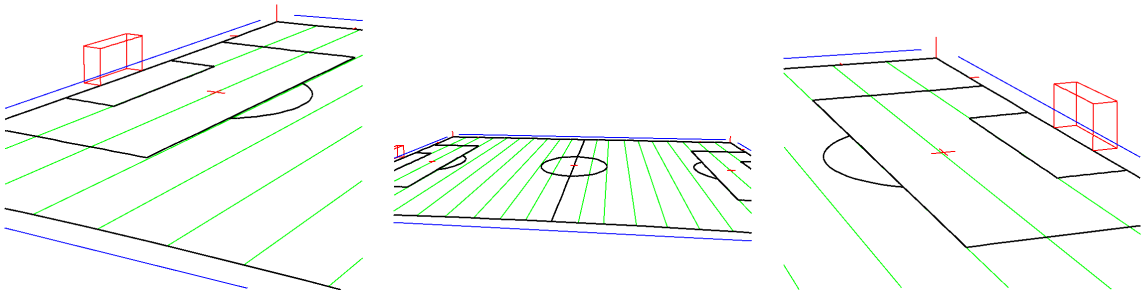


Figure 5.2 Projections of the used field model.

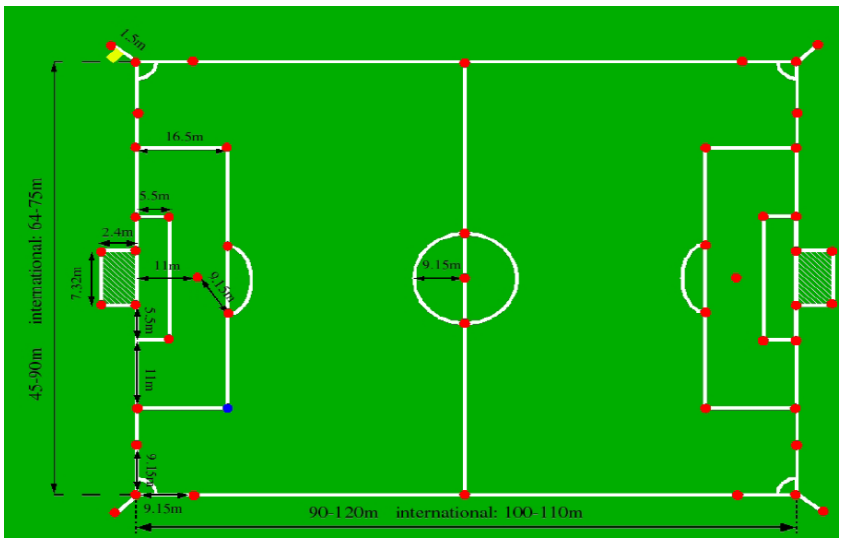


Figure 5.3 Field model with its known metrics. Taken from [Pro06]

5.1.1 Curve Model

All field curves are described by the parametric functions $\boldsymbol{w}(s), \boldsymbol{n}(s)$ and $\boldsymbol{d}(s)$ with the parameter $0 \leq s \leq 1$, where the start of the curve is given by $s = 0$ and the end by $s = 1$. For a given value of s the corresponding homogeneous 2D-point on the field plane $\boldsymbol{w}(s)$, its normal $\boldsymbol{n}(s)$ and direction $\boldsymbol{d}(s)$ can be obtained in world coordinate system (WCS). The normal is always directed to the left side of the curve (direction).

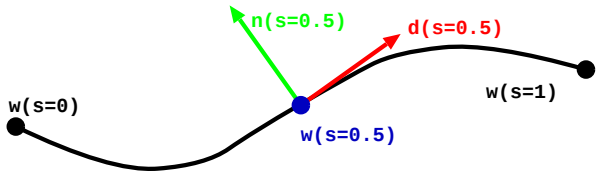


Figure 5.4 Model curve defined by the parametric functions $\boldsymbol{w}(s), \boldsymbol{n}(s)$ and $\boldsymbol{d}(s)$: The scalar s specifies an individual point, its direction and its normal on this curve.

Straight Lines

A straight line is specified by its startpoint $\boldsymbol{w}_b$ and its endpoint $\boldsymbol{w}_e$. The position of an intermediate point can be simply determined by linear interpolation between both endpoints.

$$\boldsymbol{w}(s) = \boldsymbol{w_e} \cdot s + (1 - s) \cdot \boldsymbol{w_s} \tag{5.1}$$

The direction $\boldsymbol{d}$ and normal $\boldsymbol{n}$ of the curve at point $\boldsymbol{w}(s)$ are independent of s and are given by:

$$\boldsymbol{d} = \frac{\boldsymbol{w}_e - \boldsymbol{w}_s}{\|\boldsymbol{w}_e - \boldsymbol{w}_s\|} \qquad \boldsymbol{n} = \begin{pmatrix} -\boldsymbol{d}_y \\ \boldsymbol{d}_x \\ 0 \end{pmatrix} \tag{5.2}$$

Arcs

The circle in the center of the field and the arcs at the penalty areas are described by circular arcs. A circular arc is specified by its center point $\boldsymbol{w}_c$, radius r, start angle τ and opening angle δ. Intermediate points on the arc are determined by linear interpolation of the angle.

$$\boldsymbol{w}(s) = \boldsymbol{w}_c + r \cdot \begin{pmatrix} \cos(\tau + s \cdot \delta) \\ \sin(\tau + s \cdot \delta) \\ 0 \end{pmatrix} \tag{5.3}$$

The normal and direction to the corresponding intermediate point $\boldsymbol{w}(s)$ are given by:

$$\boldsymbol{n}(s) = \begin{pmatrix} -\cos(\tau + s \cdot \delta) \\ -\sin(\tau + s \cdot \delta) \\ 0 \end{pmatrix} \qquad \boldsymbol{d}(s) = \begin{pmatrix} -\sin(\tau + s \cdot \delta) \\ \cos(\tau + s \cdot \delta) \\ 0 \end{pmatrix} \tag{5.4}$$

5.1.2 Color Model

Color information is necessary for reliable feature detection. In the context of this thesis only the field color is of interest, which is used to segment image regions which belong to the field (color segmentation) and for reliable correspondence search.

For efficiency reasons, color classes are represented by a mixture of Gaussian (MoG) in the RGB16[1] colorspace and are completely specified by a set of mean values and corresponding covariance matrices. This reduces the number of entries in the lookup table from $2^{24} = 16777216$ to $2^{16} = 65536$. A pixel can be classified to belong to the field color with a

[1]The RGB16 colorspace uses 6 bits for the green channel and 5 bits for the blue and red channel respectively.

given level of confidence by considering its Mahalanobis distance to the color class (see section 2.3.3). For multimodal distributions the Mahalanobis distance is defined to be the minimum Mahalanobis distance to all Gaussians in the color class.

In most cases the field color can be approximated with sufficient accuracy by just one Gaussian, and therefore can be determined automatically. Examples where the field color can not be represented by one Gaussian are given in figure 5.5. In these cases automatic color calibration is infeasible and must be done manually.

For efficiency reasons lookup tables for the Mahalanobis distance and the probability density are precalculated for all 65536 possible colors values. As the appearance of colors may be different for each camera, each camera is associated with its own lookup table.

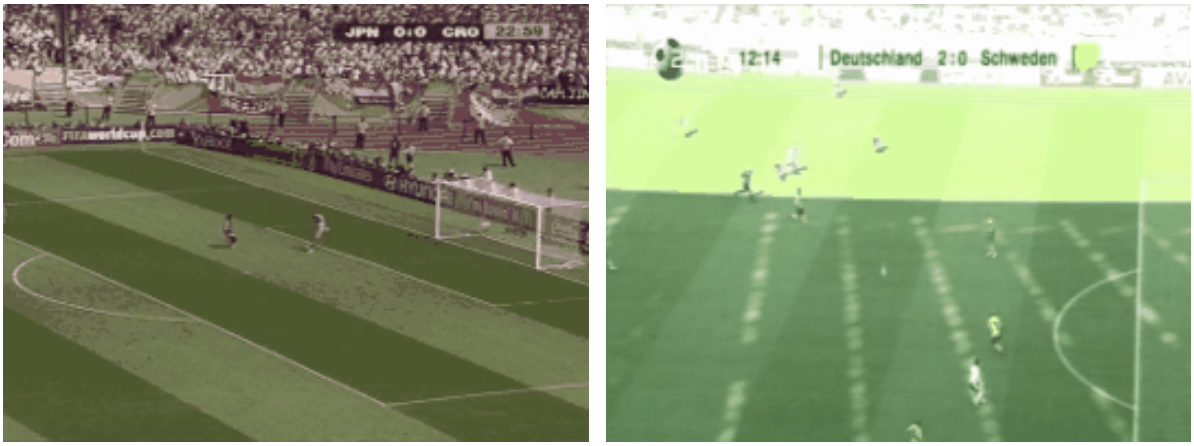


Figure 5.5 Multimodal color distribution: high contrast (left) and cast shadows (right)

5.1.2.1 Automatic Color Calibration

If the field color can be represented accurately enough by one Gaussian, it can be detected automatically from multiple images. For uncut video streams arbitrary images can be used, but in case of video streams of TV broadcasts the user must ensure that the images are from the camera of interest, as the appearance of the color may be different for each camera.

In soccer a large area of the image is covered by the field. And as the field is almost untextured (i.e. the field is almost homogeneous), we suppose that large homogeneous regions belong to the field.
The automatic color calibration is performed by following steps:

- Regions with homogeneous gray values are detected.
- Regions connected by small links are separated by morphological opening.

- Smaller regions, or regions which appear only on the image border are eliminated.
- The color values of all pixels within the remaining regions are used to determine the field color.

Detecting homogeneous regions

A pixel belongs to a homogeneous region if an inhomogeneity measure is below a threshold. In the context of this thesis, the inhomogeneity measure of a pixel is defined to be the variance of the gray values of its neighborhood. The neighborhood of a pixel is defined to be a 11×11 rectangular area (*window*) around the pixel. Figure 5.6 depicts the original image with corresponding variance image.

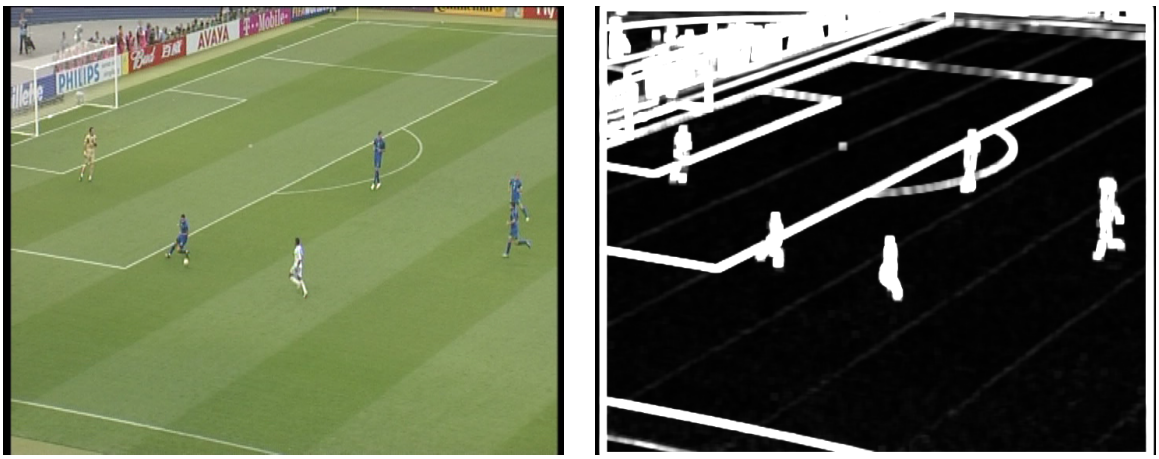


Figure 5.6 Variance image: original image (left) and variance image for a 11×11 neighborhood (right)

The variance image is determined very efficiently by using the integral images from the gray values and squared gray values. The integral images are recursively defined by:

$$\begin{aligned} I_i(x,y) &= I_i(x-1,y) + I_i(x,y-1) - I_i(x-1,y-1) + I(x,y) \\ I_s(x,y) &= I_s(x-1,y) + I_s(x,y-1) - I_s(x-1,y-1) + (I(x,y))^2 \end{aligned} \tag{5.5}$$

where $I(x,y)$ denotes the gray value of the picture element at position (x,y) in the original image. $I_i(x,y)$ and $I_s(x,y)$ denote the according entry of the integral of the gray values and integral of the squared gray values respectively. The intensity values for all pixels beyond the image border are supposed to be 0.

Using equation (2.12) to determine the variance of a pixel p at (x,y), the calculation reduces to a couple of additions, subtractions and one multiplication, as the integral images are scaled

beforehand by the factor $\frac{1}{w^2}$[2].

$$\begin{aligned}
\bar{p} &= I_i\left(x+\frac{w}{2}, y+\frac{w}{2}\right) + I_i\left(x-\frac{w}{2}-1, y-\frac{w}{2}-1\right) \\
&\quad - I_i\left(x, y-\frac{w}{2}-1\right) - I_i\left(x-\frac{w}{2}-1, y\right) \\
\bar{p^2} &= I_s\left(x+\frac{w}{2}, y+\frac{w}{2}\right) + I_s\left(x-\frac{w}{2}-1, y-\frac{w}{2}-1\right) \\
&\quad - I_s\left(x, y-\frac{w}{2}-1\right) - I_s\left(x-\frac{w}{2}-1, y\right) \\
\hat{\sigma}_p^2 &= \bar{p^2} - \bar{p}^2
\end{aligned} \tag{5.6}$$

Note that $\bar{p}$ and $\bar{p^2}$ indicate the mean of all gray values and squared gray values within the window respectively. $\hat{\sigma}_p^2$ is simply the sample variance of the gray values within the window (see equation (2.12)).

To identify pixels with homogeneous neighboorhood, the variance image is binary thresholded. Empirically, we discovered that a threshold value of 3σ, with σ being the standard deviation of the inhomogeneity values, works reliable to segment homogeneous regions. To estimate σ we select randomly 1000 entries from the variance image and determine their variance according to equation (4.143).

In the next step, a morphological erosion is performed to separate regions that are connected by small links. Then all regions which appear only on the border of the image or their areas are below a threshold are filtered out. The threshold value for the minimum area is set to 10% of the total image size. A region is defined to be at the border, if it does not intersect a rectangle of half size of the image, placed at the image center.

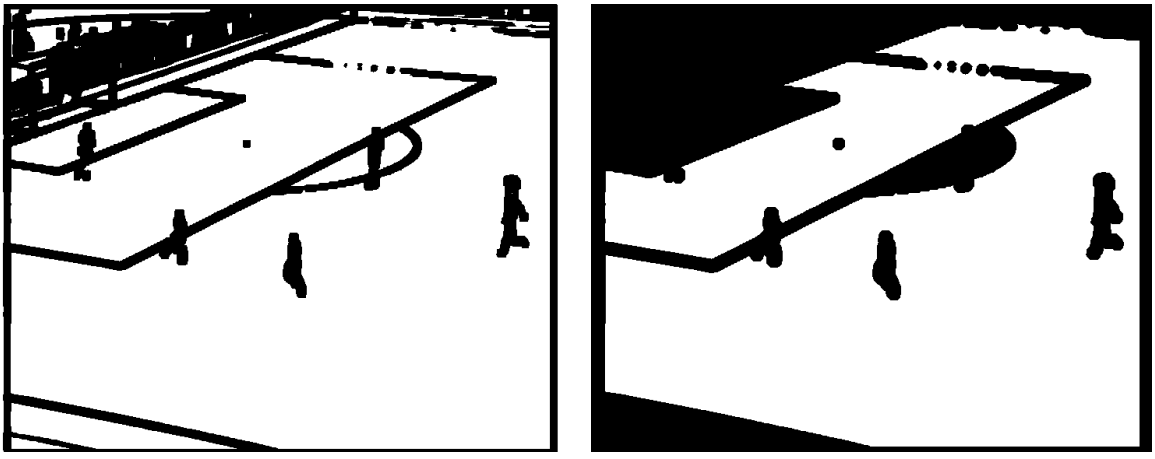


Figure 5.7 Homogeneous regions: homogeneous regions (left) and thresholded (right)

[2] w denotes the window size and is in our case 11

Finally, the color values of all pixels within the remaining regions are used as samples to determine the field color class. Figure 5.8 shows the point cloud, representing the sample color values in the RGB16 space of above example.

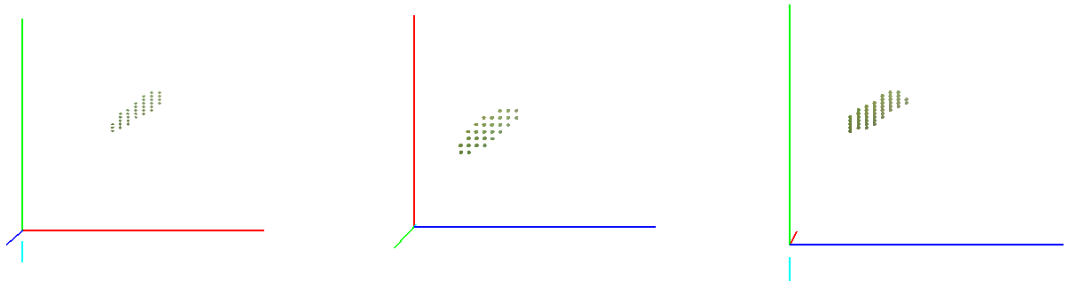


Figure 5.8 Field color samples in the RGB16 color space: red-green plane (left), blue-red plane (center) and blue-green plane (right)

Figure 5.9 shows the ellipsoid, representing the color values classified as field color in the RGB16 color space with a maximum Mahalanobis distance of 4 (confidence level of 99.9%) [3] The colored points in the point cloud indicate the occurrence of these colors in the sample set which was used to generate the color class.

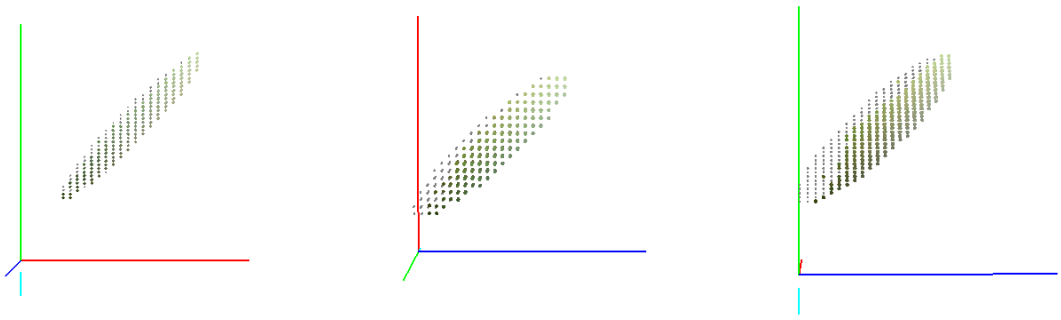


Figure 5.9 Classified colors as field color in the RGB16 color with Mahalanobis distance of 4: red-green plane (left), blue-red plane (center) blue-green plane (right)

The classification result for Mahalanobis distance of 2 (confidence level of 74.9%) can be seen in figure 5.10.

[3] This value means, that 99.9% of all colors belonging to the field color class are within this ellipsoid.

Figure 5.10 Color classified image with Mahalanobis distance of 2.

5.2 Objective Functions

Model-based state estimation determines the parameters of the model that best fits to the observation (the image). The fitness of an estimate is obtained by the objective or cost function that returns an error e. Some objective functions based on correspondences between image points and model points/lines will be presented in this section.

A correspondence D is a three tuple containing the model point/line, the corresponding image point and its uncertainty (i.e. 2×2 covariance matrix).

$$D =< \boldsymbol{w}, \boldsymbol{p}, \Sigma_{pp} > \tag{5.7}$$

The methods for finding the actual correspondences are described in the according sections in detail (see sections 5.4.1.2 and 5.6.2).

5.2.1 Objective Functions from Point-Point Correspondences

This is the only objective function providing two error values per correspondence. These are the x- and y- differences between the projected model point and the corresponding image point (see figure 5.11(a)). For each model point $\boldsymbol{w} = (w_x, w_y, w_z, 1)^T$ with proper projection $\boldsymbol{q} = (q_x, q_y, 1)^T$ and corresponding image point $\boldsymbol{p} = (p_x, p_y, 1)^T$, both error values are evaluated as

the x and y differences of $\boldsymbol{p}$ and $\boldsymbol{q}$.

$$\boldsymbol{e}_{pt} = \begin{pmatrix} p_x - q_x \\ p_y - q_y \end{pmatrix} \tag{5.8}$$

Alternatively, the distance of the 3D-model point in camera coordinate system $\boldsymbol{c}$ to the line of sight of $\boldsymbol{p}$ can be used (see figure 5.11(b)). The line of sight of a point $\boldsymbol{p}$ in camera coordinate system (CCS) is defined to be:

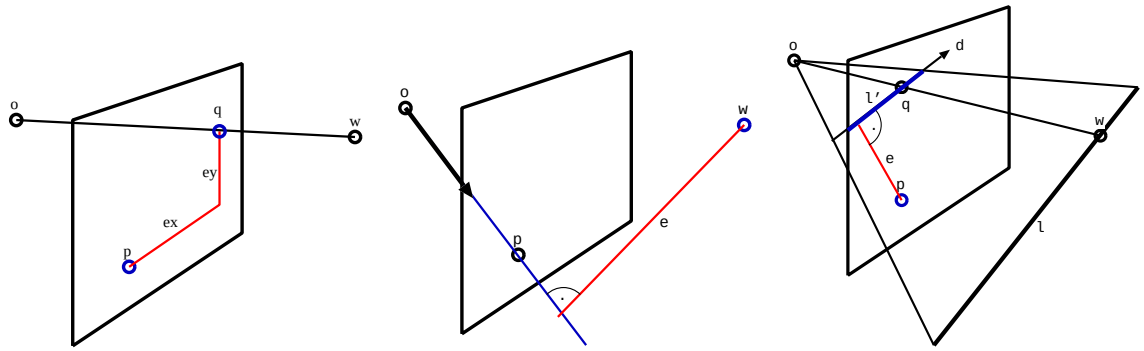
$$K^{-1}\boldsymbol{p} \tag{5.9}$$

with K being the camera matrix (see section 3.1.3) and the distance can be determined by the cross product

$$e = \frac{(K^{-1}\boldsymbol{p}) \times \boldsymbol{c}}{\|K^{-1}\boldsymbol{p}\|} \tag{5.10}$$

with $\boldsymbol{c} = \tilde{R}t \cdot \boldsymbol{w}$ (see section 3.1.1).

However, this error function provides just one error value per correspondence and therefore the minimum number of necessary correspondences increases. In the context of this thesis,



(a) (b) (c)

Figure 5.11 Objective functions: (a) error value of point-point correspondence in 2D, (b) error value of point-point correspondence in 3D and (c) error value of point-line correspondence in 2D.

the cost function $\boldsymbol{e}_{pt}$ is used in the *Constant Parameter Estimation* module and in the *prediction stage* of the *Iterative Camera Calibration* module. The advantages of $\boldsymbol{e}_{pt}$ compared to e are twofold. First, as mentioned above $\boldsymbol{e}_{pt}$ provides 2 error values per correspondence. This

halves the number of required correspondences. Secondly, the uncertainties are obtained from the image and given in pixel coordinate system. Therefore, they can be directly used as the uncertainties of the corresponding error values.

5.2.2 Objective Functions based on 3D model lines

Two types of lines are considered in the context of this thesis, model edges and model lines. As model lines are reduced to their medial axis, the differentiation between both line types is only important for the determination of the corresponding image point and its uncertainty (see section 5.5.1).

Each model line and image point correspondence provides only one error value. Let $\boldsymbol{w}$ be a model point and $\boldsymbol{q}$ its projection on the image. Furthermore, let $\boldsymbol{d}$ be the normalized projection of the direction of the curve at the model point $\boldsymbol{w}$. The error value is given by the distance of the image point $\boldsymbol{p}$ to the tangent defined by $\boldsymbol{q}$ and $\boldsymbol{d}$ (see figure 5.11(c)).

$$e_{ln} = \| \boldsymbol{d} \times (\boldsymbol{p} - \boldsymbol{q}) \| \tag{5.11}$$

As $\boldsymbol{d}$ and $(\boldsymbol{p} - \boldsymbol{q})$ are vectors and their homogeneous components equal 0, their cross product is perpendicular to the image plane and its length is equivalent to its z component. So, the above equation can also be written as:

$$e_{ln} = d_x \cdot (p_y - q_y) - d_y \cdot (p_x - q_x) \tag{5.12}$$

For straight lines, the tangent coincides with the projection of the line itself, but the determination of the tangent is necessary for arcs.

Alternatively, the distance can be determined by the use of the homogeneous representation of the tangent $\boldsymbol{l}'$. As the direction vector $\boldsymbol{d}$ is also the vanishing point of the tangent, both $\boldsymbol{d}$ and $\boldsymbol{q}$ lie on the tangent and uniquely define it. The tangent $\boldsymbol{l}'$ is obtained by a simple cross product as follows:

$$\begin{aligned} \boldsymbol{l}' &= \boldsymbol{d} \times \boldsymbol{q} \\ &= \begin{pmatrix} d_y \\ -d_x \\ d_x \cdot q_y - d_y \cdot q_x \end{pmatrix} \end{aligned} \tag{5.13}$$

The error value is then given by the scalar product

$$e_{ln} = \boldsymbol{l}'^T \boldsymbol{p} \tag{5.14}$$

which again results in equation (5.12). Note, that the error value is the distance of point $\boldsymbol{p}$ to the tangent $\boldsymbol{l}'$ scaled by the normal length of the tangent, which is given by its first two components. As we suppose to have a normalized direction vector $\boldsymbol{d}$, the normal of the tangent has unit length, and the error value e_{ln} is given by the distance.

5.3 Constant Parameter Estimation

Constant parameter estimation is done once at the beginning of each game. All relevant parameters which are assumed to be constant during the game are determined using multiple images taken from the camera of interest. Constant parameters are camera position $\breve{\boldsymbol{t}}_c$, roll angle β, principal point (C_x, C_y), aspect ratio[4] θ, radial distortion coefficient κ and field size (F_l, F_w). The best results are achieved by using multiple images of both penalty areas and the center circle with different focal lengths (particularly all corners of the field should be visible).

The user must provide correspondences between image points and model points. Supported model points are the intersection points of field markings, the endpoints of the posts of the goals, and the tips of the corner flags. All supported model points are printed as red dots in figure 5.3.

Additionally, the user can provide a covariance matrix for the image points of the correspondences. This is visually done by defining the uncertainty ellipse around the selected image point. The resulting covariance matrices are determined from these ellipses [Pro06]. In example depicted in figure 5.12, the uncertainty ellipses are specified as the red circles with same radius, thus the overall covariance matrix is homoscedastic.

The actual calibration process for given point correspondences is two-staged. In the first stage, closed form solutions for most parameters are determined by making some reasonable assumptions. Finally, these parameters are used as an initial estimate for the nonlinear least squares solution using the weighted Levenberg-Marquardt method as described in sections 4.2.4 and 4.2.4.1.

The idea is very similar to standard calibration methods, but the determined parameter sets differ. Standard methods use a calibration pattern with known metrics and determine the con-

[4] Actually the pixel width is estimated. However, since the pixel height is provided by the user, we will present the results in terms of the aspect ratio.

stant but unknown intrinsic parameters of the camera. They assume, that the extrinsic parameters are dynamic, so the multiple views of the pattern are taken from different poses. However, we split the parameters into two different sets where some of the intrinsic and some of the extrinsic are supposed to be constant. Additionally, we use the markings of the field instead of the calibration pattern and have to estimate additionally some model parameters (field size) as well.

5.3.1 Initial Estimation

The Levenberg-Marquardt estimation is a nonlinear iterative method and requires an initial estimate, which is determined by a closed-form solution in projective space [Zha00].

As model points depend on the estimated parameters (field size) and can not be estimated simultaneously in this initial stage, the field size is fixed to the mean value of $105m \times 68.5m$ [FIF07]. If the field size differs considerably from this mean value, it has to be provided by the user.

The initial parameters are estimated in two steps. First, all views with a sufficient number of correspondences are used to determine the camera's position and its roll angle. Then the dynamic camera parameters (pan, tilt and zoom) are determined for all views, by using the estimated parameters from previous step.

5.3.1.1 Camera Position and Roll Angle Estimation

The camera projection matrix (see equation (3.11)) describes the transformation of points in the world coordinate system (WCS) to points in the pixel coordinate system (PCS). As all points lying on the field have a zero z component, the transformation is independent of the third column of the rotation matrix and can be written in homogeneous coordinates as:

$$\boldsymbol{p} \simeq \underbrace{K\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t}\right)}_{H} \boldsymbol{w}' \tag{5.15}$$

with $\boldsymbol{t} = -\breve{R}_c \breve{\boldsymbol{t}}_c$, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ being the first and second column vector of the camera rotation matrix $\breve{R}_c$ respectively and $\boldsymbol{w}' = (w_x, w_y, 1)^T$. Note, that above equation is defined up to a non zero scale factor λ, indicated by the projective equality $\simeq$. The resulting projection matrix H maps homogeneous points on the field plane to homogeneous points on the image and is called *planar Homography*.

$$(\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3) = \lambda K\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{t}\right); \tag{5.16}$$

with $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ being the three column vectors of H.

Given at least four non-collinear[5] point correspondences of field points for each image, the planar homography can be determined as described in [HZ03].

From each homography, two constraints on the intrinsic camera parameters can be derived by the orthonormality property of the rotation matrix.

By multiplying K^{-1} on the left side, we get the following first and second column vectors:

$$K^{-1}\boldsymbol{h}_1 = \lambda \boldsymbol{r}_1 \tag{5.17}$$

$$K^{-1}\boldsymbol{h}_2 = \lambda \boldsymbol{r}_2 \tag{5.18}$$

Taking the square on both sides results in:

$$\boldsymbol{h}_1^T \omega \boldsymbol{h}_1 = \lambda^2 \boldsymbol{r}_1^T \boldsymbol{r}_1 = \lambda^2 \tag{5.19}$$

$$\boldsymbol{h}_2^T \omega \boldsymbol{h}_2 = \lambda^2 \boldsymbol{r}_2^T \boldsymbol{r}_2 = \lambda^2 \tag{5.20}$$

The matrix $\omega = K^{-T}K^{-1}$ is called *Image of the Absolute Conic* (IAC) [HZ03] and has the following structure for zero skew.

$$\omega = \begin{pmatrix} \frac{1}{f_x^2} & 0 & -\frac{c_x}{f_x^2} \\ 0 & \frac{1}{f_y^2} & -\frac{c_y}{f_y^2} \\ -\frac{c_x}{f_x^2} & -\frac{c_y}{f_y^2} & \frac{c_x^2}{f_x^2} + \frac{c_y^2}{f_y^2} + 1 \end{pmatrix} \tag{5.21}$$

Since the scaling factor λ is unknown, we can derive the following two equations:

$$\boldsymbol{h}_1^T \omega \boldsymbol{h}_1 = \boldsymbol{h}_2^T \omega \boldsymbol{h}_2 \tag{5.22}$$

$$\boldsymbol{h}_1 \omega \boldsymbol{h}_2 = 0 \tag{5.23}$$

At least two planar homographies are required to determine all entries of ω and thus to determine the missing intrinsic parameters of the camera [Zha00].

Unfortunately, the focal length of the camera may be different for each image. This leads to only two linear equations and we need additional constraints to reduce the number of unknowns. Naturally, we set the principal point to the image center as this is its expected position and has the least impact on the projection [KH90, KH94]. Theoretically, this should be sufficient to determine f_x and f_y, but the equation system is ill-conditioned, and the determined

[5] non-collinear means, that none three of them lie on a line, thus are not collinear

results are infeasible in most cases. By additionally providing the aspect ratio (e.g. using the camera specification, or setting the aspect ratio to 1, i.e. $S_x = S_y$ (see equation (3.13))), leads to an overdetermined equation system, and a reliable estimate of the last unknown intrinsic parameter, the focal length f, can be determined by least squares.

Since the only remaining unknown is now the focal length f, the camera matrix can be represented by equation (3.13) and all point features are transformed beforehand by the inverse of K_a. All homographies are determined by these new point features and the camera matrix simplifies to $K := K_p$. This leads to the following IAC:

$$\omega = \begin{pmatrix} \frac{1}{f^2} & 0 & 0 \\ 0 & \frac{1}{f^2} & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.24}$$

and equations (5.22) and (5.23) respectively become:

$$\frac{h_{11}^2 + h_{21}^2}{f^2} + h_{31}^2 = \frac{h_{12}^2 + h_{22}^2}{f^2} + h_{32}^2 \tag{5.25}$$

$$\frac{h_{11} \cdot h_{12} + h_{21} \cdot h_{22}}{f^2} + h_{31} \cdot h_{32} = 0 \tag{5.26}$$

leading to the following least square solution:

$$f = \sqrt{\frac{\left(h_{11}^2 + h_{21}^2 - h_{12}^2 - h_{22}^2\right)^2 + \left(h_{11} \cdot h_{12} + h_{21} \cdot h_{22}\right)^2}{\left(h_{11}^2 + h_{21}^2 - h_{12}^2 - h_{22}^2\right) \cdot \left(h_{32}^2 - h_{31}^2\right) + \left(h_{11} \cdot h_{12} + h_{21} \cdot h_{22}\right) \cdot \left(-h_{31} \cdot h_{32}\right)}} \tag{5.27}$$

After determining the focal length, the unknown scaling factor λ is simply found by the geometric mean as follows:

$$\begin{aligned} \lambda &= \sqrt{\lambda_1 \cdot \lambda_2} \\ \lambda_1 &= \frac{\|K^{-1}\boldsymbol{h}_1\|}{\|\boldsymbol{r}_1\|} = \|K^{-1}\boldsymbol{h}_1\| \\ \lambda_2 &= \frac{\|K^{-1}\boldsymbol{h}_2\|}{\|\boldsymbol{r}_2\|} = \|K^{-1}\boldsymbol{h}_2\| \end{aligned} \tag{5.28}$$

The rotation matrix is determined as:

$$R = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3)\,; \tag{5.29}$$

with

$$\boldsymbol{r}_1 = \frac{K^{-1}\boldsymbol{h}_1}{\lambda_1} \qquad \boldsymbol{r}_2 = \frac{K^{-1}\boldsymbol{h}_2}{\lambda_2} \qquad \boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$

Due to noise, R may not satisfy the orthogonality property of a rotation matrix. In a further step, the final rotation matrix $\breve{R}_c$ is obtained by the best orthonormal matrix in terms of the smallest Frobenius norm to R [Zha00].

Finally, the rotation angles α, β and γ are determined using equation (2.7) and the translation vector is given by:

$$\breve{\boldsymbol{t}}_c = -\frac{1}{\lambda}\breve{R}_c^T K^{-1}\boldsymbol{h}_3 \tag{5.30}$$

If more than one view with sufficient correspondences is given, the initial values for the camera position $\breve{\boldsymbol{t}}_c$ and roll angle β are determined by the average.
The average of angles $\delta_i, i = 1 \ldots \tilde{j}$ is obtained by:

$$\bar{\delta} = \operatorname{atan2}\left(\sum_{i=1}^{\tilde{j}} \cos\left(\delta_i\right), \sum_{i=1}^{n} \sin\left(\delta_i\right)\right) \tag{5.31}$$

with $\tilde{j}$ being the number of views with sufficient point correspondences. However, taking the average of the position and the roll angle makes the dynamic parameters become inconsistent with the so obtained position and roll angle. Therefore, the dynamic parameters are estimated again in the next step.

5.3.1.2 Dynamic Parameters Estimation

Knowing the camera position $\breve{\boldsymbol{t}}_c$ and the roll angle β, the dynamic parameters can be estimated by just two point correspondences. However, if more than two correspondences are available, we use all combinations to calculate the parameters and take the average. First we determine the focal length as the geometric mean of all combinations of two point correspondences. After the focal length is determined, one point correspondence is sufficient to determine the tilt and the pan angle of the camera. As at least two point correspondences are available, we determine the angles for all point correspondences and take the average by (5.31).

Estimating Focal Length from Two Point Correspondences

The focal length is determined from two point correspondences as follows. Let $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ be two model points in world coordinate system (WCS) and $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ their corresponding image points in pixel coordinate system (PCS) respectively.
Then

$$\begin{aligned} \breve{\boldsymbol{v}}_1 &= \frac{\breve{\boldsymbol{w}}_1 - \breve{\boldsymbol{t}}_c}{\|\breve{\boldsymbol{w}}_1 - \breve{\boldsymbol{t}}_c\|} \\ \breve{\boldsymbol{v}}_2 &= \frac{\breve{\boldsymbol{w}}_2 - \breve{\boldsymbol{t}}_c}{\|\breve{\boldsymbol{w}}_2 - \breve{\boldsymbol{t}}_c\|} \end{aligned} \tag{5.32}$$

are the normalized vectors from the camera center to $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ respectively. The points $\breve{\boldsymbol{w}}_1$ and $\breve{\boldsymbol{w}}_2$ are the Euclidean counterparts of the homogeneous points $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ respectively.
The angle φ between both vectors is then given by:

$$\cos(\varphi) = \breve{\boldsymbol{v}}_1^T \breve{\boldsymbol{v}}_2 \tag{5.33}$$

For a given image point $\boldsymbol{p}$, the vector in camera coordinate system (CCS) describing the line of sight for this point can be determined by:

$$\boldsymbol{q} = K^{-1}\boldsymbol{p} \tag{5.34}$$

Thus, if the camera matrix is known, the angle φ can also be obtained from the corresponding image points as follows:

$$\begin{aligned} \cos(\varphi) &= \frac{\boldsymbol{q}_1^T \boldsymbol{q}_2}{\|\boldsymbol{q}_1\|\|\boldsymbol{q}_2\|} \\ &= \frac{\boldsymbol{p}_1^T K^{-T} K^{-1} \boldsymbol{p}_2}{\sqrt{\boldsymbol{p}_1^T K^{-T} K^{-1} \boldsymbol{p}_1}\sqrt{\boldsymbol{p}_2^T K^{-T} K^{-1} \boldsymbol{p}_2}} \\ &= \frac{\boldsymbol{p}_1^T \omega \boldsymbol{p}_2}{\sqrt{\boldsymbol{p}_1^T \omega \boldsymbol{p}_1 \boldsymbol{p}_2^T \omega \boldsymbol{p}_2}} \end{aligned} \tag{5.35}$$

Setting equations (5.35) and (5.33) equal, the focal length can be obtained as the solution of:

$$\frac{\boldsymbol{p}_1^T \omega \boldsymbol{p}_2}{\sqrt{\boldsymbol{p}_1^T \omega \boldsymbol{p}_1 \boldsymbol{p}_2^T \omega \boldsymbol{p}_2}} = \breve{\boldsymbol{v}}_1^T \breve{\boldsymbol{v}}_2 \tag{5.36}$$

The solution is given by:

$$f^2 = \frac{2 \cdot (d^2 \cdot a \cdot b - c^2)}{2 \cdot c - d^2 \cdot (a+b) + \sqrt{(d^2 \cdot (a+b) - 2 \cdot c)^2 - 4 \cdot (d^2 \cdot a \cdot b - c^2)(d^2 - 1)}} \tag{5.37}$$

with

$$a = \boldsymbol{p}_1^T \boldsymbol{p}_1 \qquad b = \boldsymbol{p}_2^T \boldsymbol{p}_2$$
$$c = \boldsymbol{p}_1^T \boldsymbol{p}_2 \qquad d = \breve{\boldsymbol{v}}_1^T \breve{\boldsymbol{v}}_2$$

Estimating Pan and Tilt Angle from Single Point Correspondence

After the focal length is determined, the camera matrix is known and the projection can be written as:

$$\begin{aligned} \boldsymbol{p} &= \lambda K \breve{R}_c \breve{\boldsymbol{v}} \\ K^{-1}\boldsymbol{p} &= \lambda \breve{R}_c \breve{\boldsymbol{v}} \\ \boldsymbol{q} &= \lambda \breve{R}_c \breve{\boldsymbol{v}} \end{aligned}$$

The unknown scale factor λ can be removed by normalizing the left hand side, as the rotation on the right hand side does not affect the length of $\breve{\boldsymbol{v}}$.

$$\boldsymbol{q}' = \breve{R}_c \breve{\boldsymbol{v}} \qquad \text{with} \qquad \boldsymbol{q}' = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}$$

As $\boldsymbol{q}'$, $\breve{\boldsymbol{v}}$ and the roll angle β are known, the pan angle γ can be determined using the first row of the rotation matrix $\breve{R}_c$ (see section 2.2.1) as follows:

$$\begin{aligned} \cos(\beta) \cdot \cos(\gamma) \cdot v_x - \cos(\beta) \cdot \sin(\gamma) \cdot v_y + \sin(\beta) \cdot v_z &= q'_x \\ \underbrace{-\cos(\beta) \cdot v_y}_{a} \cdot \sin(\gamma) + \underbrace{\cos(\beta) \cdot v_x}_{b} \cdot \cos(\gamma) + \underbrace{\sin(\beta) \cdot v_z - q'_x}_{c} &= 0 \end{aligned} \tag{5.38}$$

Reordering the terms and taking the square on both sides result in a quadratic equation.

$$\begin{aligned} -a \cdot \sin(\gamma) &= b \cdot \cos(\gamma) + c \\ a^2 \cdot \left(1 - \cos^2(\gamma)\right) &= b^2 \cdot \cos^2(\gamma) + 2 \cdot b \cdot c \cdot \cos(\gamma) + c^2 \\ 0 &= \left(a^2 + b^2\right) \cdot \cos^2(\gamma) + 2 \cdot b \cdot c \cdot \cos(\gamma) + c^2 - a^2 \end{aligned} \tag{5.39}$$

Both solutions for $\cos(\gamma)$ are given by:

$$\cos_{1,2}(\gamma) = \frac{-b \cdot c \pm a \cdot \sqrt{a^2 + b^2 - c^2}}{a^2 + b^2} \tag{5.40}$$

and the four solutions for γ are given by:

$$\begin{aligned} \gamma_1 &= \arccos(\cos_1(\gamma)) & \gamma_2 &= 2 \cdot \pi - \gamma_1 \\ \gamma_3 &= \arccos(\cos_2(\gamma)) & \gamma_4 &= 2 \cdot \pi - \gamma_3 \end{aligned} \tag{5.41}$$

As only two solutions meet the above equation (5.38), the remaining two solutions can be eliminated. However, equation (5.38) just considers the x-coordinate of the rotated vector $\breve{\boldsymbol{v}}$. By considering the z-coordinate as well, we can eliminate the last wrong solution as it lies behind the camera, i.e. its z-coordinate is negative.

To determine z, we suppose that the roll angle is 0 and the tilt angle is $\frac{\pi}{2}$, i.e. the horizon goes through the image center and is parallel to the x-axis of the image. This reduces the calculation efforts for the z-coordinate, and the constraint is given by:

$$\sin(\gamma) \cdot v_x + \cos(\gamma) \cdot v_y > 0 \tag{5.42}$$

The angle satisfying above constraint is the unique solution for the pan angle γ.

The tilt angle α is determined in an analogous manner, by the use of the second row of the rotation matrix. Both solutions for $\cos(\alpha)$ are also given analogously to (5.40) with

$$\begin{aligned} a &= \sin(\beta) \cdot \cos(\gamma) \cdot v_x - \sin(\beta) \cdot \sin(\gamma) \cdot v_y - \cos(\beta) \cdot v_z \\ b &= \sin(\gamma) \cdot v_x + \cos(\gamma) \cdot v_y \\ c &= -q'_y \end{aligned} \tag{5.43}$$

The three wrong solutions are eliminated in the same manner by determining the y and the z-coordinates of the rotated vector $\boldsymbol{v}$.

Figure 5.12 shows the initial estimates for four images with used correspondences (red circles). The projected field models depicted in figure 5.12 show the accuracy of the initial estimates.

Figure 5.12 Constant Parameter Estimation: Initial estimates determined from point correspondences (red dots)

5.3.2 Final Estimation Using Least Squares

After finding the initial estimate, the final parameters are found by the weighted Levenberg Marquardt method by using the objective function from equation (5.8) (see section 4.2.4).

The parameter vector $\boldsymbol{x}$ can be split into two groups. The first group are the constant parameters, which are shared by all views, and the second group contains the dynamic parameters, which are view specific. The constant parameters are the position of the camera $\breve{\boldsymbol{t}}_c = (t_x, t_y, t_z)$, the roll angle β, the aspect ratio θ, the principal point (C_x, C_y), the radial distorion coefficient κ and the field size (F_l, F_w). The dynamic or view specific parameters are the pan angle γ, the tilt angle α and the focal length f. Thus, for j views the state vector $\boldsymbol{x}$ has the dimension $n = 3j + 10$.

It is possible to manually fix some constant parameters to predefined values and exclude them from the estimation process. In this case we get less unknowns and the estimation of the

remaining parameters gets more robust. This is sometimes done for the principal point or the field size, if the estimation process is performed with a moderate number of views.

The accuracy and robustness can be improved by specifying also correspondences which do not lie on the field plane. The only known model points which do not lie on the field plane are the corners of the posts of the goals. The tips of the corner flags can be used as well, but as their height is not specified completely by the official regulations [FIF07], these are not very reliable features.

Although the order of the parameters in the parameter vector is not important, we separate them for better understanding as shown below.

$$\boldsymbol{x} = \left(\Phi_0^T | \Phi_1^T, \ldots, \Phi_j^T\right)^T \tag{5.44}$$

with

$$\Phi_0 = (t_x, t_y, t_z, \beta, \theta, C_x, C_y, \kappa, F_l, F_w)^T$$
$$\Phi_i = (\alpha_i, \gamma_i, f_i)^T \qquad \text{for} \qquad 1 \leq i \leq j$$

Let suppose the view i contains m_i point correspondences. Then the measurement vector $\boldsymbol{z}_i$ is $2m_i$ dimensional containing the coordinates of the image points $\breve{\boldsymbol{p}}_i$. As the measurements within a view are supposed to be independent, the $2m_i \times 2m_i$ covariance matrices Σ_i have a diagonal block structure, containing the 2×2 covariance matrices of the image points.

The measurements of each view are independent as well, and the covariance matrix of the stacked measurement vector

$$\boldsymbol{z} = \left(\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, \ldots, \boldsymbol{z}_j^T\right)^T \in \mathbb{R}^m \tag{5.45}$$

has also a diagonal structure containing 2×2 matrices.

$$\Sigma_{zz} = \operatorname{diag}\left(\Sigma_1, \Sigma_2, \ldots, \Sigma_j\right) \in \mathbb{R}^{m \times m} \tag{5.46}$$

with $m = \sum_{i=1}^{j} 2m_i$

The task is now to find the nonlinear least squares solution $\hat{\boldsymbol{x}}$ given the measurement vector $\boldsymbol{z}$, its covariance matrix Σ_{zz} and the observation model h.

$$\boldsymbol{z} = h\left(\boldsymbol{x}\right) \tag{5.47}$$

$h\left(\boldsymbol{x}\right)$ contains the projections of the model points for the parameters $\boldsymbol{x}$ and the entries of $\boldsymbol{z}$ contain the corresponding image points. Note that if the field size is estimated simultaneously,

the model points depend on the field size and $h(\boldsymbol{x})$ is not just the projection of a constant model point.

The final estimate $\hat{\boldsymbol{x}}$ is determined using algorithm 4.5 with the modification as described in section 4.2.4.1, and the initial estimate $\boldsymbol{x}_0$ is provided by the previous step (see section 5.3.1)

In the lines 6 and 14 of algorithm 4.5, the Jacobian of h is required. We determine the partial derivatives from the Jacobian by the difference quotient as described in [Pro06]. Using the difference quotient leads in general to slower convergence, but as this method is applied in the pre-game stage it is not time critical.

The structure of the $m \times n$ Jacobian is depicted in (5.48).

$$J = \left(\begin{array}{c|ccc} \boxed{\frac{\partial h_1}{\partial \Phi_0}} & \boxed{\frac{\partial h_1}{\partial \Phi_1}} & & \\ \boxed{\frac{\partial h_2}{\partial \Phi_0}} & & \boxed{\frac{\partial h_2}{\partial \Phi_2}} & \\ \vdots & & & \ddots \\ \boxed{\frac{\partial h_j}{\partial \Phi_0}} & & & \boxed{\frac{\partial h_j}{\partial \Phi_j}} \end{array}\right) \tag{5.48}$$

The $m_i \times 3$ matrices $\boxed{\frac{\partial h_i}{\partial \Phi_i}}$ denote a block in the Jacobian for view i, that depends only on the dynamic parameters Φ_i. The block of size $m_i \times 10$ matrices $\boxed{\frac{\partial h_i}{\partial \Phi_0}}$ represents view i, which depends on the constant parameters Φ_0. From the structure of the Jacobian one can see the connection between all views over the constant parameters, and it is obvious that the more views available the more accurate the constant parameters can be estimated. The structure of the Jacobian is comparable to the Jacobian obtained by bundle adjustment problems (see [HZ03, TMHF00]).

Figure 5.13 shows the resulting final estimates of the example shown in figure 5.12. The estimated parameters are given in table 5.1.

t_x(m)	t_y(m)	t_z(m)	β(rad)	θ	C_x(px)	C_y(px)	$\kappa(\frac{1}{m^2})$	F_l(m)	F_w(m)
-1.23	-72.44	17.53	0.00169	1.0679	370.06	225.32	-111.20	105.07	68.29

Table 5.1 Results for the constant parameter estimation example

Figure 5.13 Constant Parameter Estimation: Final estimates determined from point correspondences (red dots) using the initial estimate (see figure 5.12).

5.3.2.1 Quality of the Estimated Parameters

The covariance matrix of the estimated parameters is determined by the last iteration of the Levenberg-Marquardt method and is given by equation (4.123).

The required covariance matrices for the image points can be specified by the user by providing the uncertainty ellipses for these points. However, as this is an intuitive way to specify a covariance matrix, we assume that the overall covariance matrix Σ_{zz} is known only up to an unknown scale factor σ_w^2, which is estimated by equation (4.100). The covariance matrix of the estimate is given by

$$\Sigma_{\hat{x}\hat{x}} = \hat{\sigma}_w^2 \left(J^T \Sigma_{zz}^{-1} J \right)^{-1} \tag{5.49}$$

5.3.3 Experimental Results

The method described in this section was evaluated successfully in a game between Germany and Switzerland, where 9 calibrated cameras covering the whole field (see figure 5.14) were used to evaluate our method by comparing it with ground truth data.

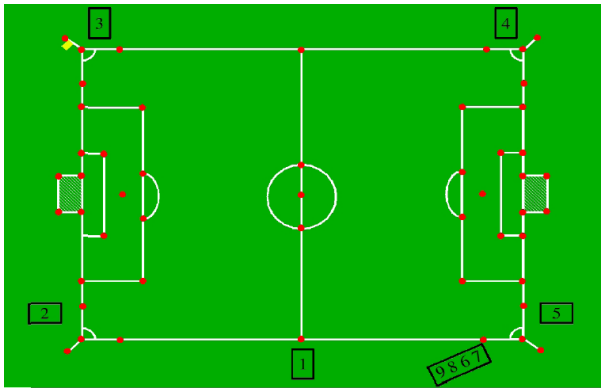


Figure 5.14 Camera configuration for the game Germany vs. Switzerland. Taken from [Pro06]

The distances of all corner points and the center point of the field to all other points, including the camera locations, were obtained by a laser measuring device with a centimeter class accuracy. The field size could be obtained directly from these measurements, whereas the position of the cameras were determined by trilateration [Tor02].

The ground truth data for the intrinsic camera parameters was determined by a standard calibration method using multiple images of a calibration pattern. For control purpose, this was done before and after the game. However, no ground truth data was available for the orientation of the cameras. While the pan and tilt angle are varying parameters, the only constant parameter without any groundtruth data was the roll angle β. Furthermore, all the measurements were obtained at the same height (approx. 1.7m above the field) resulting in unreliable outcomes of the trilateration problem in terms of z coordinates of the cameras. In order to determine the precise ground truth values, z-coordinates and the roll angles were adjusted manually from the captured video streams.

Camera Configurations

All the cameras had fixed intrinsic parameters and since the calibration process could not consider the non-zooming cameras, the geometric mean of all the focal lengths from each view was devised to find the focal length.

Besides rotating cameras, we have also used several stationary cameras (4 and $6-9$). For

the stationary cameras the estimation of the field size was not possible as only a small portion of the field was visible. Furthermore, the simultaneous estimation of the principal point and the radial distortion coefficient was not reliable, as only one view was available. Therefore, we have fixed the principal point to the image center and estimated only the remaining parameters.

Another special configuration were the cameras $[6-9]$. As they did not considerably differ in their position (just a few centimeters), we treated all four cameras as an additional virtual camera. The views from each of the four cameras were supposed to be captured by the virtual camera with different focal length and orientation. As their intrinsic parameters were not identical, we have compared just the estimated position and field size for this virtual camera.

Tables 5.2-5.11 present the results for all configurations, where bold entries in the ground truth data indicate manually adjusted and possibly unreliable ground truth data and bold entries in the estimated parameters indicate fixed values during the estimation.

The position and field size are given in meters, whereas the focal length is given in millimeters, and the distortion coefficient κ is given in $\frac{1}{m^2}$.

Camera 1	t_x	t_y	t_z	f	θ	C_x	C_y	κ	F_l	F_w
Ground Truth	4.4	-76.6	***8.6***	21.56	1.0798	343.2	274.7	1054	104.3	68.0
known fields.	4.5	-76.5	8.6	19.81	1.0569	355.7	93.0	2966	***104.3***	***68.0***
unkn.fields.	4.3	-72.6	8.2	18.66	1.0490	354.9	201.2	2261	100.0	68.5

Table 5.2 Experimental results for camera 1 (rotating)

Camera 2	t_x	t_y	t_z	f	θ	C_x	C_y	κ	F_l	F_w
Ground Truth	-98.9	-20.1	***7.5***	20.58	1.0951	355.1	288.6	420	104.3	68.0
known fields.	-99.5	-20.0	7.3	20.72	1.0934	373.7	290.7	1307	***104.3***	***68.0***
unkn.fields.	-97.2	-20.0	7.3	20.335	1.0961	377.8	294.7	1268	102.0	68.2

Table 5.3 Experimental results for camera 2 (rotating)

Camera 3	t_x	t_y	t_z	f	θ	C_x	C_y	κ	F_l	F_w
Ground Truth	-56.8	67.9	***7.3***	25.03	1.0823	417.4	328.1	54	104.3	68.0
known fields.	-56.8	66.0	6.7	23.76	1.0501	288.2	240.9	263	***104.3***	***68.0***
unkn.fields.	-57.8	66.2	6.7	23.87	1.0520	295.8	249.6	258	106.3	68.2

Table 5.4 Experimental results for camera 3 (rotating)

Camera 5	t_x	t_y	t_z	f	θ	C_x	C_y	κ	F_l	F_w
Ground Truth	91.0	-34.9	***7.3***	18.76	1.0830	364.9	294.2	635	104.3	68.0
known fields.	90.4	-35.3	7.3	18.85	1.0881	374.5	273.4	1193	***104.3***	***68.0***
unkn.fields.	90.9	-35.4	7.3	18.83	1.0894	374.3	274.4	1163	104.8	68.2

Table 5.5 Experimental results for camera 5 (rotating)

Camera 4	t_x	t_y	t_z	f	θ	κ	C_x	C_y
Ground Truth	54.5	67.7	***7.3***	18.63	1.1028	549	363.1	283.9
Estimated	57.1	65.7	6.9	17.90	1.0784	67	***359.5***	***287.5***

Table 5.6 Experimental results for camera 4 (stationary)

Camera 6	t_x	t_y	t_z	f	θ	κ	C_x	C_y
Ground Truth	33.8	-72.3	***12.4***	12.80	1.1055	617	349.3	288.0
Estimated	34.1	-67.3	11.3	12.31	1.1036	1844	***359.5***	***287.5***

Table 5.7 Experimental results for camera 6 (stationary)

Camera 7	t_x	t_y	t_z	f	θ	κ	C_x	C_y
Ground Truth	33.5	-72.3	***12.4***	9.22	1.1050	-583	355.7	314.6
Estimated	33.7	-71.1	11.9	9.37	1.1315	-2430	***359.5***	***287.5***

Table 5.8 Experimental results for camera 7 (stationary)

Camera 8	t_x	t_y	t_z	f	θ	κ	C_x	C_y
Ground Truth	33.3	-72.3	***12.4***	12.44	1.1096	1789	342.4	293.6
Estimated	33.9	-71.7	12.3	12.95	1.1255	1099	***359.5***	***287.5***

Table 5.9 Experimental results for camera 8 (stationary)

Camera 9	t_x	t_y	t_z	f	θ	κ	C_x	C_y
Ground Truth	33.0	-72.3	***12.4***	17.39	1.0858	1110	361.9	302.5
Estimated	31.4	-69.3	12.0	17.04	1.0847	413	***359.5***	***287.5***

Table 5.10 Experimental results for camera 9 (stationary)

Virtual Camera 10	t_x	t_y	t_z	F_l	F_w
Ground Truth	33.4	-72.3	***12.4***	104.3	68.0
Estimated (known field size)	33.6	-70.8	12.3	***104.3***	***68.0***
Estimated (unknown field size)	34.0	-71.6	12.3	105.2	67.8

Table 5.11 Experimental results for the virtual camera (quasi-rotating)

5.4 Scene Iterator

The *Scene Iterator* module successively searches for sequences, which are captured from the camera of interest, by iterating through all sequences. This is done by iterating over the video stream, and identifying new video segments at shot boundaries. Each video segment is then classified with two independent classifiers to *fieldview* or *non fieldview* and to *slow motion replay* or *non slow motion replay*. All video segments, which are classified as *non fieldview* or *slow motion replay* are skipped, as they have no value for the tracking task. The former do not contain any suitable information for the tracker and the latter contains sequences which are out of time line.

All other sequences are potentially from the camera of interest and are passed to the *Initial Parameter Estimation* module, which determines initial values for the dynamic camera parameters using extracted image lines.

The task of the Scene Iterator module is to detect the beginning of each sequence and to identify fieldviews and slow motion replays. The methods used are discussed in the proceeding subsections.

5.4.1 Shot Boundary Detection

A *shot* is an uninterrupted sequence of frames from one camera and *shot boundaries* are the transitions between two subsequent shots.

In soccer broadcasts, one can differentiate between three types of shot boundaries: *cuts*, *dissolves* and *effects*.

A *Cut* is an abrupt transition, i.e. within two subsequent frames.

A *Dissolve* appears, when the frames from the previous shot fade out, while the frames from the next shot fade in. In soccer broadcasts, a dissolve takes usually $15 - 20$ frames and frames within the transition show one image from the previous shot superimposed on another image of the next shot.

Effects wipe out one shot by another shot and are usually used in soccer broadcasts at the beginning and end of replay blocks.

Figure 5.15 shows samples for shot cuts, dissolves and effects from the final game of the FIFA World Cup 2006.

Figure 5.15 Shot boundary examples: first row shows two shot cuts, second row shows a dissolve and last row shows an effect

The most important shot boundary in the context of this thesis is the *cut*, as it is the only shot boundary used during the active game, i.e. if the game is not interrupted or no replays are shown.

5.4.1.1 Related Work

Shot boundary detection methods usually determine some quantity that describes the degree of dissimilarity between two subsequent frames or frames within a range. The majority of the methods use pixel differences, statistical differences, histogram comparisons, edge comparisons, compression differences and motion vectors.

The simplest method to detect a shot boundary between two frames is to determine the number of pixels which differ in their gray values considerably [ZKS01]. Usually, this is done after smoothing the images to suppress the effects of noise and small camera motions. A pixel is supposed to be different if a measure is above a threshold, and a shot boundary between

both frames is detected if the number of differing pixels is above a second threshold. Hampapur et al. [HWJ94] use the ratio between the change of the gray values and the gray values of the second image, which is called *chromatic image*. Unfortunately methods based on pixel differences are very sensitive to camera and object motion and therefore infeasible for soccer broadcasts.

Another approach is to compare statistical measures which are determined from color values, instead of the color values directly. Katsuri et al. [KJ91] partition the image in several regions and compare a statistical measure which is based on the mean and standard deviation of the gray values within these regions. Nagasaka et al. [NT92] also divide the image into several regions, but use hypothesis testing on the color histograms of these regions. The images are divided into 16 equal parts and only the 8 best matching regions are used to overcome camera and object motion and noise.

The most common method is the *Histogram Comparison*. The idea behind histogram comparison is, that the histograms of subsequent frames within one shot probably are almost the same, while the histograms of frames in two different shots probably differ considerably. Small camera and object motions have in most cases only a small influence to the histogram. Swanberg et al. [SSJ93] use the difference in the gray level histogram, whereas Ueda et al. [UMY91] use the difference in the color histogram change rate to detect shot boundaries.

Compression Difference methods use the change in the size of JPEG compressed images or in MPEG compressed videos as an indicator for shot boundaries [LAF+93, AHC94]. However, JPEG compression requires too much computational time and the use of precomputed information in MPEG video streams is only advantageous if using MPEG compressed video streams.

Feature-based methods compare the number and the position of extracted image features in both frames [ZMM95]. These methods are relatively robust against camera and object motion and achieve a better detection rate than histogram methods. However, these methods are slow as usually edges and corners are used, and therefore have to be extracted in each image.

Motion-based methods [NPZ02, UMY91, ZKS01, Sha95] determine motion vectors by block matching techniques to identify camera and object motion. They are usually used in combination with pixel-based methods to reduce false positives caused by camera and object motion.

Good surveys and comparisons of shot boundary detection are given in [YCK98, BR96, JHEJ98].

5.4.1.2 Our Approach

In the context of this thesis, a hybrid method combining two different techniques is used to detect shot cuts as well as effects and dissolves. They are designed to detect shot boundaries in sport broadcasting reliably, in particular in soccer, but are also tested successfully for movies and commercials.

We use a feature-based method to detect shot cuts, and color histogram comparison to detect dissolves and effects. After a shot boundary is detected, we disable shot boundary detection for the next 20 frames to suppress multiple detections.

Detecting Shot Cuts

As mentioned above, we detect shot cuts from image features. The main idea is, that within a shot most image features stay visible and only disappear because of object motion or camera motion. After a shot cut however, almost all features disappear. A shot cut is found if the detection rate of image features between two subsequent images drops rapidly.

Unlike other feature-based methods, we do not consider the position, orientation or shape of features, thus we do not limit the camera or object motion. As long as sufficient features can be tracked between two subsequent images, i.e. the images are neither too blurry nor the camera or object motion is so fast that two subsequent images show different views of a scene, the shot cut detection works reliably.

As mentioned above, feature-based shot boundary detection are slow if features have to be extracted in both images and compared to each other. However, we follow another idea which is much faster. Instead of using corners or edges, we use randomly selected point features in the previous frame I_{t-1}. To decide whether the points are visible in the current frame I_t, we track these points using the Lucas Kanade tracker [LK81], which uses a block matching method. This is efficiently done using search at different scales with the help of *Pyramid Images* [Bou00].

However, tracking points in one direction can cause high outlier rates and we use the idea suggested by Hartley et al. [Har97], which will be explained in the following.

First a set of points are tracked from frame I_{t-1} to frame I_t. Let S_{t-1} be the subset of successful tracked points $\boldsymbol{q}$ from frame I_{t-1}, and S_t be the corresponding set of points $\boldsymbol{p}$ from frame I_t. Then the points $\boldsymbol{p}$ from S_t are back tracked to points $\boldsymbol{q}'$ in frame I_{t-1}. Let the set of successfully back tracked points be denoted by S'_{t-1}. Finally, points in S_{t-1} with a larger than 2 pixels offset distance with respect to their back tracked correspondence in S'_{t-1} are eliminated, since these feature points are not reliable.
Figure 5.16 shows the tracking process of a single point.

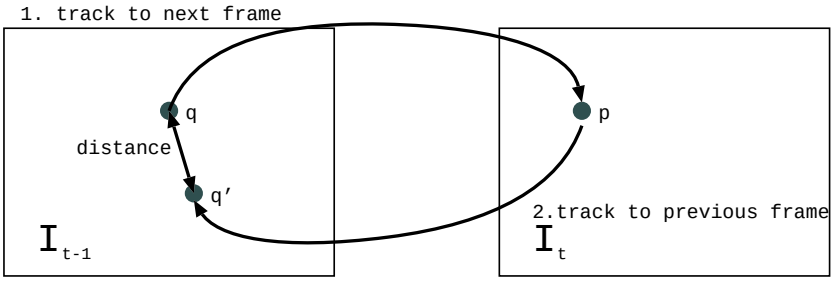


Figure 5.16 Point feature detection

The problem of wrong point correspondences still exists, particularly in image regions with similar texture, but this method improves the results considerably by eliminating most outliers.

After finding point correspondences, we determine the ratio $\xi_t = \frac{|S'_{t-1}|}{|S_{t-1}|}$ which is called point tracking rate hereafter. It is the ratio between the number of found correspondences $|S'_{t-1}|$ and the total number of initial points $|S_{t-1}|$. Figure 5.17 shows the point tracking rate for the corresponding sequences of both shot cuts shown in the first row of figure 5.15. The spikes indicate the shot boundary for these sequences.

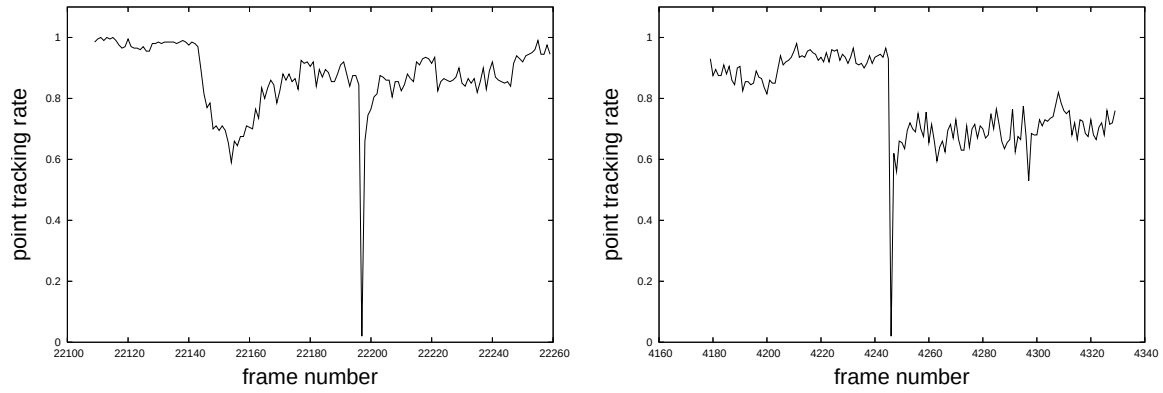


Figure 5.17 Point tracking rates for both shot cuts from the first row of figure 5.15. Spikes indicate shot cuts.

A shot cut is detected, if the point tracking rate falls below a given threshold, while it is above a second dynamic threshold before or after the current time step. The rule for shot cut detection is given by following heuristic:

$$cut = \begin{cases} true & \text{if} \quad \xi_t < 0.16 \wedge \left(\frac{\xi_{t-1}}{\xi_t} > 6 \vee \frac{\xi_{t+1}}{\xi_t} > 6 \right) \\ false & otherwise \end{cases} \tag{5.50}$$

This rule has been proven to provide reliable results during our empirical evaluation.

Detecting Dissolves and Effects

Dissolves and *Effects* are detected using a color histogram comparison. Unlike a shot cut the transition of dissolves and effects are not abrupt, but usually take about $15 - 20$ frames in soccer broadcasts. As the histograms of two subsequent images within a dissolve or effect differ little, we compare the histograms of the current frame I_t at time t with the histogram of frame I_{t-10} from time $t - 10$. Using a distance of 10 frames is reasonable, as no smaller shot exists, thus can not be missed, while most dissolves considerably differ in their histograms within 10 frames.

However, fast camera motion or object motion can also cause a fast change in the histograms. For this reason, we buffer the last 20 histogram comparison results, and assume a shot cut if within these 20 values at least 16 values lower than a given threshold exist.

For efficiency reasons, we use the color histogram of a considerably reduced colorspace. We use only an 8 bit representation of colors in the RGB color space, where the green channel takes 4 bits while the red and blue channel take 2 bits each.

Since normalized histograms (sum of all entries is one) are statistical distributions of color values, they can be compared by the so-called *Bhattacharyya* distance [Bha43]. The Bhattacharyya distance is given by:

$$d_B = \sum_{i=0}^{256} \sqrt{h_1(i) \cdot h_2(i)} \tag{5.51}$$

with h_1 and h_2 being the normalized histograms of two images. It can be interpreted as the scalar product of two vectors having as components the square roots of the probability of the color values. Geometrically it is the cosine of the angle between these two vectors.
Figure 5.18 shows the histograms for the second and the third row of figure 5.15 respectively.

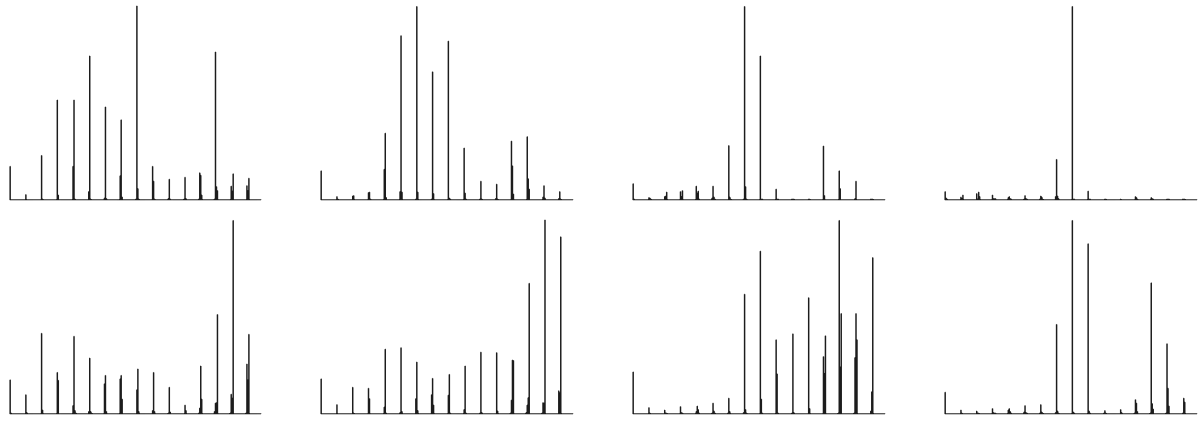


Figure 5.18 Histograms for the second and the third row of figure 5.15 respectively.

Figure 5.19 shows, the histogram similarity measure for the two sequences of figure 5.15. Intervals with small values indicate shot boundaries.

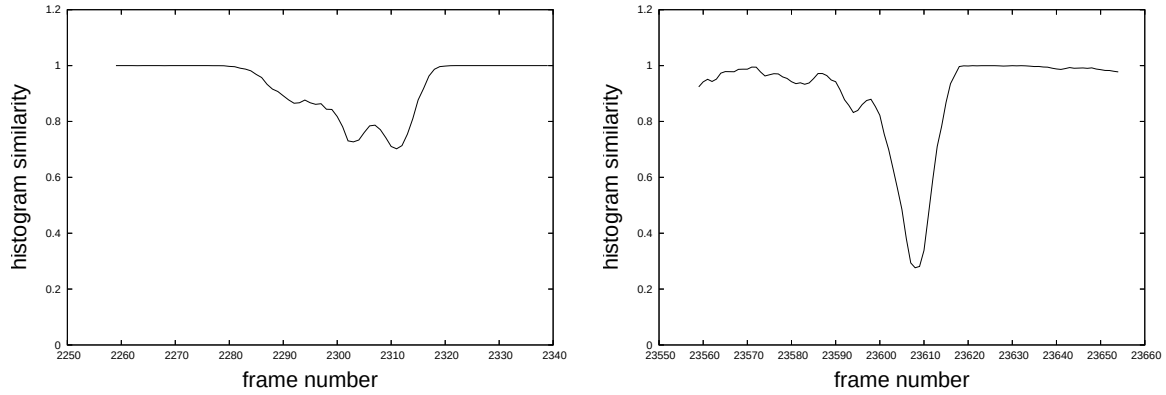


Figure 5.19 Histograms similarity measure over both sequences of the second (left) and third (right) row of figure 5.15 respectively.

5.4.1.3 Experimental Results

We have successfully applied our method to several TV broadcasts of soccer games.

For evaluation, we ran our method on the first 38000 frames ($\sim$ 25 minutes) of the final game of the FIFA World Cup 2006 and compared the results with manually obtained data. We have manually identified 155 shot cuts and 72 dissolves and effects.

The evaluation results are given in *recall* and *precision* rates, which is common in the context of shot boundary detection [BR96, PS97]. Let n_c, n_f and n_m be the number of correct, false (false positives) and missed boundaries (false negatives) respectively. Then *recall* and *precision* rates are defined as:

$$recall = \frac{n_c}{n_c + n_m} \tag{5.52}$$

$$precision = \frac{n_c}{n_c + n_f} \tag{5.53}$$

The precision and recall rates for the test case are given in table 5.12

The algorithm was tested on an AMD Athlon 64 machine with 2GB RAM running at 2200 MHz, and needed 11.7 milliseconds on average, which is equivalent to 85.5 frames per second. However, due to overhead in the image capturing (≈ 15 milliseconds) the overall method runs at a speed of 37 frames per second on average, which is still realtime.

	#	false positives	false negatives	precision	recall
shot cuts	155	1	0	99.36%	100.00%
dissolves & effects	72	4	9	94.03%	87.50%
overall	227	5	9	97.76%	96.04%

Table 5.12 Precision and recall rates for shot boundary detection

5.4.2 Fieldview Detection

Out of field views as well as *close up* views have no value for the tracking task and are skipped. The *Fieldview Detection* determines therefore whether a sequence shows the field and thus probably contains valuable information or not.

It analyzes 10 frames by taking every fifth image at the beginning of a video segment. If at least 8 fieldviews are detected within these 10 selected frames, the sequence is classified as a fieldview sequence. Otherwise the video segment is rejected and the Scene Iterator searches for the next video segment.

To decide if a given image shows the field or not, the number of pixels classified as field color is counted within a specific region in the image: the region of interest (ROI). The upper and lower part of the image are not reliable indicators for fieldviews, as for example the upper part of the image contains in most cases the stands. Therefore, we do not use the upper third and lower fifth of the image.

Even if the number of pixels classified as field color is sufficient to detect out of field views, the detection of close up views is unreliable. Therefore, the proposed method has two stages. In the first stage, the number of pixels within the ROI which are classified as field color is determined. If this number is below 67% of the total ROI size, the image is classified as non-fieldview.

Otherwise, the second stage determines the field region from the classified pixels as described in section 5.5.1.1. Then the ratio between the field region size and the number of not classified pixels within the field region is calculated, i.e. the relative size of holes in the field region. If this second ratio is above 20%, the image is probably a close up view and is therefore rejected. Above threshold values are obtained by empirical evaluation and have shown to work reliable. Figure 5.20 shows some examples for field view frames (including a false positive), whereas figure 5.21 shows examples for non-fieldviews (including a false negative).

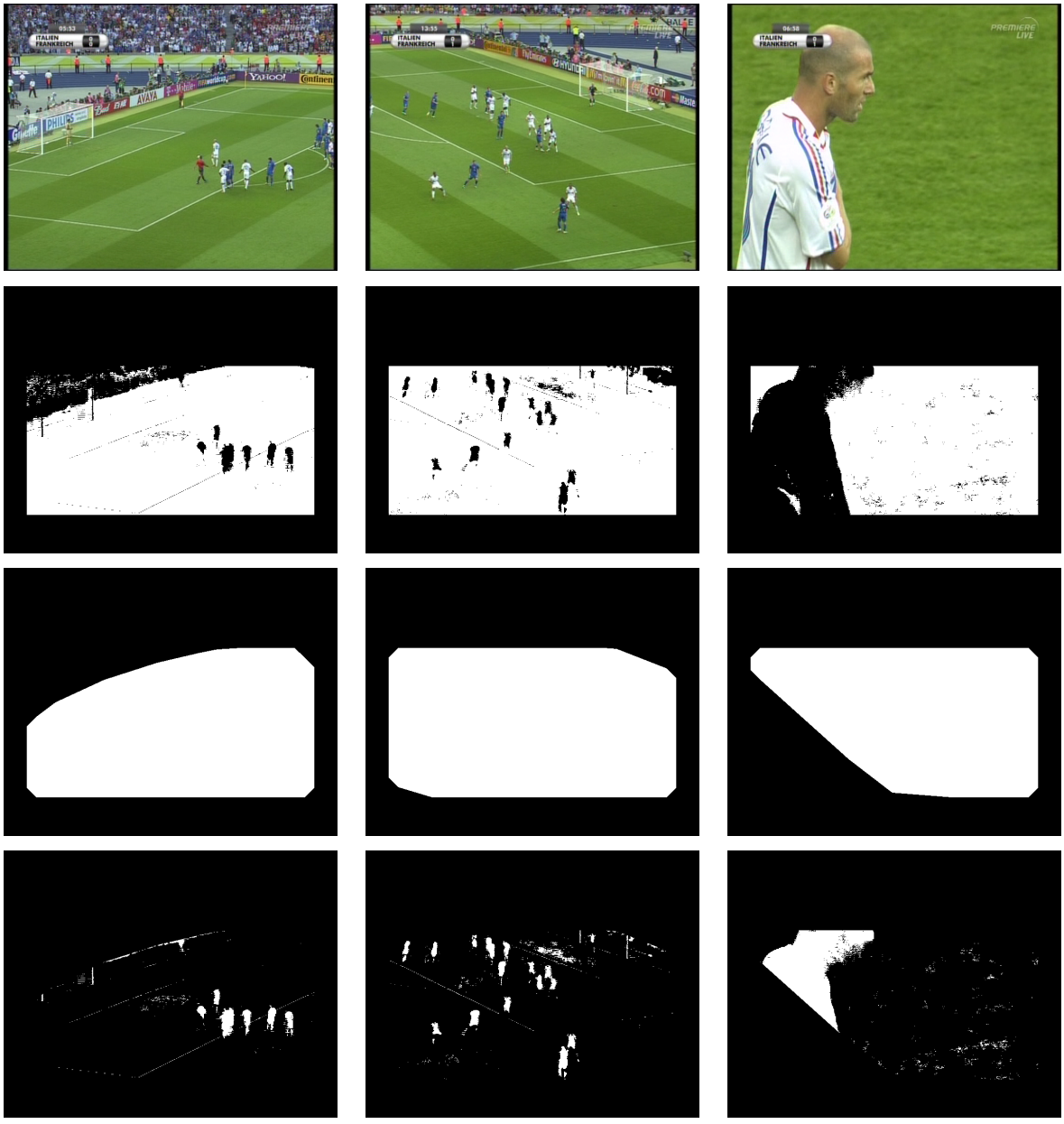


Figure 5.20 Examples for fieldviews. From top to bottom: original image, color classified image, extracted field region and holes within field region. Right column shows a sample for a false positive

Figure 5.21 Non-Fieldviews: Examples for close up view (left), out of field view (right) and false negative (right).

5.4.2.1 Experimental Results

We have tested the field view detection on sequences of several soccer games from the FIFA World Cup 2006. We determined the precision and recall rates for single frames as well as for sequences after shot boundaries with manually obtained ground truth data. The results are shown in tables 5.13 and 5.14 respectively.

game	frames	false positives	false negatives	precision	recall
Italy - France	12000	402	16	96.75%	99.87%
Germany - Costa Rica	6000	133	65	97.81%	98.92%
Czech Republic - Ghana	6000	139	26	97.73%	99.57%

Table 5.13 Precision and recall rates for field view detection in a single frame

game	frames	seq.	fieldviews	false pos.	false neg.	precision	recall
Italy - France	14000	102	41	1	2	99.01%	98.04%
Germany - C.Rica	12000	71	26	1	1	98.59%	98.59%
Cz. Rep. - Ghana	12000	72	26	3	0	96.00%	100.00%

Table 5.14 Precision and recall rates for field view detection for sequences

Most false positives came from close up views or sequences captured behind the goals, where most false negatives were sequences showing corner kicks. Former cases occur due to wrong field segmentation results or due to the large areas being visible through the nets of the goals.

5.4.3 Slow Motion Detection

In soccer broadcasts, crucial events are often shown in a replay block immediately after they occur, or during smaller breaks. These replay blocks consist of several video segments, which show the event from different perspectives and often include slow motion sequences from the camera of interest. As slow motion replays are out of time line, these sequences are of no value for the tracker and are filtered out.

Slow motion sequences are generated from recordings of standard cameras by repeating several frames to reduce the framerate, or from highspeed cameras. As recordings of highspeed cameras are mainly used for slow motion replays, we do not consider these cameras and limit our method to standard cameras.

5.4.3.1 Related Work

Several approaches for slow motion replay detection exist that are working in the compressed and spatial domain.

Kobla et al. [KDD99] use the bitrate information and macroblock motion in the compressed domain of MPEG videos to identify slow motion replays. However, we deal with general video sources, thus we do not want to constrain ourselves to a specific video format.

Pan et al. [PBS01] use a *Hidden Markov Model* (HMM) to detect the whole replay block, containing editing effects and multiple slow motion sequences. They use the fact that important events are replayed from several perspectives and thus, in most cases multiple sequences occur during these replays including at least one slow motion sequence. Furthermore, they assume that each replay begins and ends with an editing effect (see figure 5.15). The features which are used for the HMM are determined from the image difference and histogram comparison of subsequent frames.

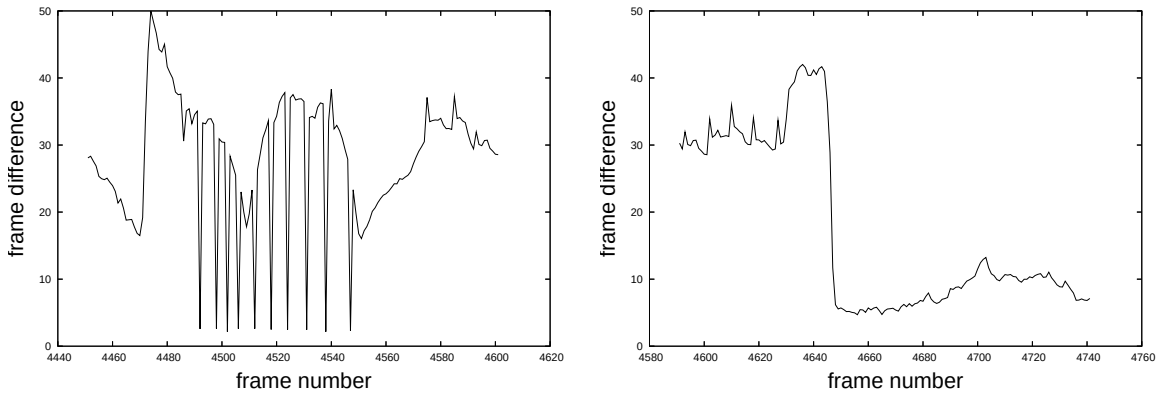
However in our case, where the *Fieldview* detection and the following *Initial Parameter Estimation* reject most sequences which do not match the camera of interest, it is sufficient to identify single sequences as slow motion segments rather than to identify whole replay blocks.

5.4.3.2 Slow Motion Sequence Detection

We use a modified version of the so-called *zero crossing measure* which was proposed in [PBS01], as it makes a good trade-off between efficiency and reliability.

Unlike sequences showing static scenes, slow motion sequences have strong fluctuations in differences between subsequent frames. These fluctuations arise from the repeated frames. Unlike the frame differences of two non-repeated frames which are mainly caused by object and camera motion, the frame differences of two repeated frames have their origin in noise which is caused by transmission or recording. Thus, the frame differences during a slow motion sequence have strong spikes which indicate repeated frames.

Let I_{t-1} and I_t be two subsequent monochrome images and $D_t\left(x,y\right) = I_t\left(x,y\right) - I_{t-1}\left(x,y\right)$ their elementwise difference. Then, the difference measure d_t is given by the standard deviation of all intensity values of D_t.
Figure 5.22 shows frame differences for slow motion and non-slow motion sequences.



(a) (b)

Figure 5.22 Frame difference for (a) slow motion and (b) non slow motion sequences.

It can be seen in figure 5.22(b), that the absolute values of the interimage differences are not good indicators. The step in the right diagram indicates a shot cut from a dynamic sequence to a highly static sequence.

To differentiate between slow motion replays and static scenes, we do not only consider the frame difference d, but also take the absolute gradient of this measure into account. The absolute gradient g is defined to be:

$$g_t = |d_{t-1} - d_t| \tag{5.54}$$

Figure 5.23 shows the absolute gradient of the examples in 5.22 respectively.

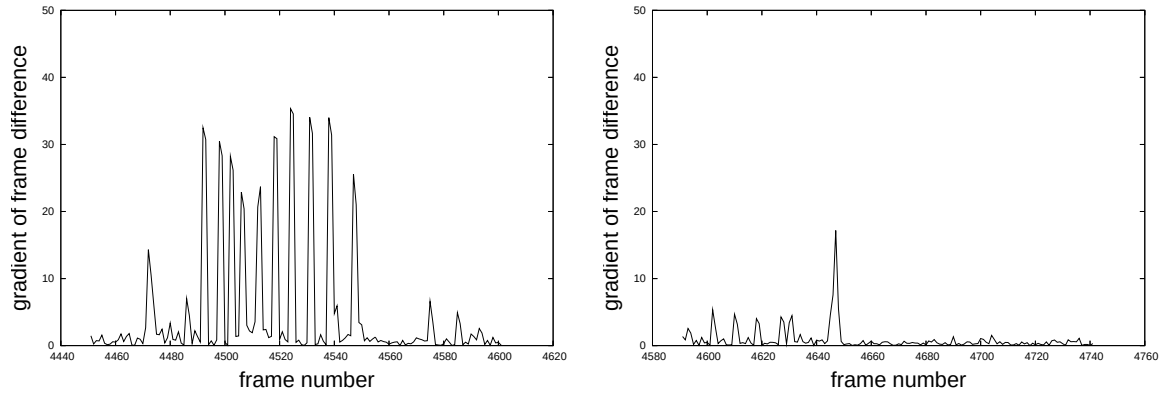


Figure 5.23 Absolute Gradient of Frame Difference for Slow Motion and Non Slow Motion sequences for above example 5.22

One would expect two peaks in the gradient diagram for a corresponding peak in the difference diagram. However, the peaks in the difference diagram have a width of only one frame, thus both peaks in the corresponding gradient diagram fuse to a single peak of width 2.

Repeated frames can be detected as frames where the absolute gradient value is above a given threshold δ_1, while the difference measure is below a second threshold δ_2. A slow motion replay is detected if at least three repeated frames within a sliding window are found. In the context of this thesis, a window size of 50 frames and the threshold values $\delta_1 = 4.8$ and $\delta_2 = 4.0$ have been shown to generate reliable results.

5.4.3.3 Experimental Results

The slow motion detector was tested for the whole final game of the FIFA World Cup 2006 including the overtime. We have manually classified 33 slow motion replays for the regular playing time and 9 for overtime.

Table 5.15 shows the evaluation of the slow motion detector.

# frames	seq.	false positives	false negatives	precision	recall
202587	1148	0	3	100%	99.74%

Table 5.15 Precision and recall rates for slow motion detection

5.5 Initial Parameter Estimation

The *Initial Parameter Estimation* module determines the initial dynamic parameters of the camera of interest at the beginning of each new sequence. These are needed as an entry point for the automated tracking of the camera as described in section 5.6. The beginning of a sequence is specified by the user or it is obtained by the *Scene Iterator* module.

The parameters are determined from extracted field lines without the need to identify the corresponding model lines, using only the knowledge of the camera position. This means that if the sequence is from the camera of interest, an initial estimate can be determined without knowing which model line corresponds to a given image line. However, it is necessary that a sufficient number of field lines are visible in the image. Therefore, the initial parameter estimates can only be determined if the goal or penalty areas are visible.

Even if enough field lines could be detected, problems may still arise due to noise, and the estimated parameters may be out of the *convergence range*[6] of the tracker.

Furthermore, it is not assured that the sequence is originated from the camera of interest. Such cases will result in wrong parameters due to the false assumption about the camera position. For these reasons, the image lines are used to confirm the determined parameters or to reject the current image.

For example, critical goal scenes are usually replayed from different perspectives immediately after they happen. In such a case, a sufficient number of field lines is visible, and the parameters can probably be determined. However, as the sequence is from a different camera, the obtained parameters are inconsistent to the observed image lines, and the image is rejected.

To summarize, an image is rejected if it is from a video segment which was taken from a different camera, or if the determined parameters are far from the true state. To overcome the latter case, the parameter determination is repeated for different frames until a valid parameter set could be found, or we obtained 20 invalid parameter sets. In such a case we suppose that the scene is captured by a different camera and reject it. Based on our experiences we argue that it is usually sufficient to ensure that the invalid parameters do not originate from noisy data. If a valid parameter set could be found within the 20 tested frames, the parameters are passed to the Iterative Camera Calibration (ICC) to start the tracking of the camera. If the detected parameters are from a frame other than the first in the sequence, the ICC tracks the camera in both directions starting at that frame.

The *Initial Parameters Estimation* uses the two vanishing points extracted from visible image lines corresponding to horizontal and vertical model lines for a geometric determination

[6] see section 5.6

of the camera's focal length as well as pan and tilt angle.

5.5.1 Field Line Extraction

As mentioned above, the initial parameters are estimated from extracted image lines. Since the radial distortion is known beforehand, the images are first rectified to get straight image lines and to remove the effects of lens distortion. The field line extraction can be summarized by the following steps

- First, the field region is determined by the use of field color information.
- All possible line candidate pixels are segmented as brighter pixels within the field region.
- A Hough transformation on the line candidate pixels is performed, resulting in line segments.
- Line segments which are close to each other are merged to a single line segment, by a weighted least squares.
- The merged line segments are grouped into two groups by clustering lines according to their common intersections, the vanishing points.

5.5.1.1 Extracting Line Segments

In a first step, a binary image is determined from color classification with Mahalanobis distance of 2, noise is removed by morphological erosion and the field region is determined by the convex hull (see figure 5.24 and figure 5.25(a)).

Then, field line candidates are segmented as brighter pixels within the field region (see figure 5.25(b)). A pixel is used as a line pixel candidate if its gray value is above $\bar{g} + 3 \cdot \sigma_g$, where $\bar{g}$ and σ_g are the mean and standard deviation of the gray values of the field color respectively[7].

[7]Confidence level of 99.9%

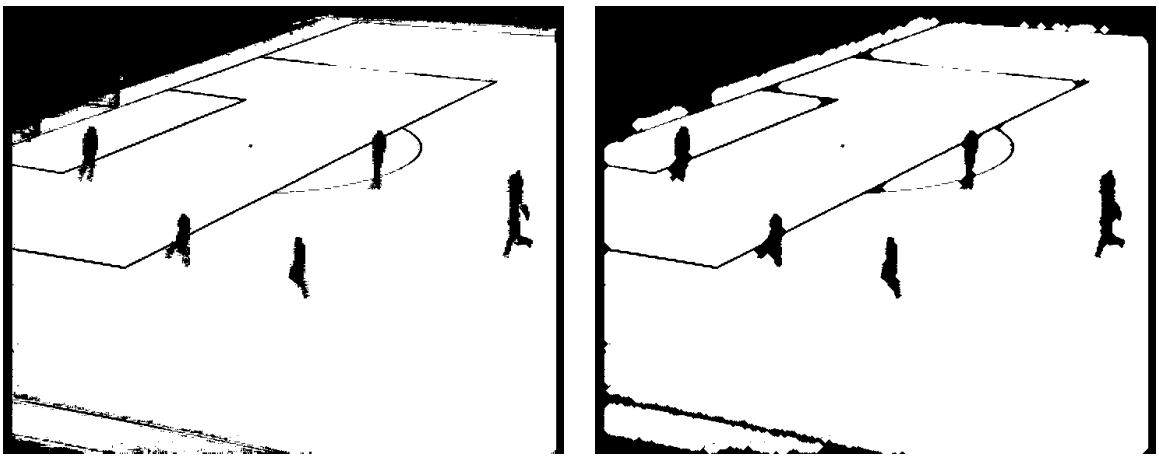
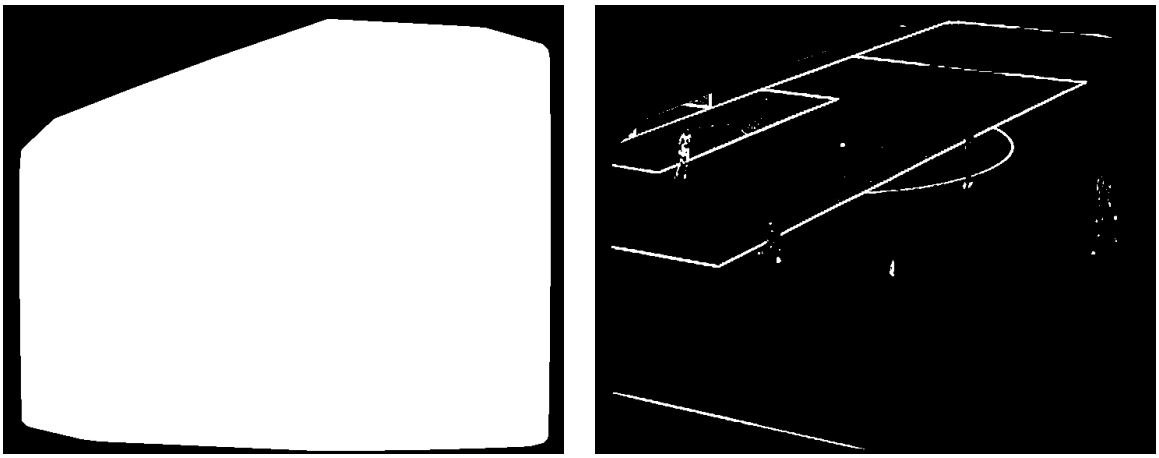


Figure 5.24 Color classified with Mahalanobis distance 2 (left), after removing noise with morphological erosion (right).



(a) (b)

Figure 5.25 Line segmentation: (a) convex fieldregion and (b) field line candidates

Finally, line segments are obtained by the Hough transformation and are given by their endpoints. Figure 5.26 shows 36 line segments obtained by the Hough transformation.

5.5.1.2 Merging Line Segments

The accuracy of the Hough transformation depends highly on the discretization levels of the state space. The state space used in the Hough transformation is the angle between the line and the x-axis, and the line's distance to the origin of the image. To obtain accurate line segments, we use one degree resolution for the angle and one pixel for the distance.

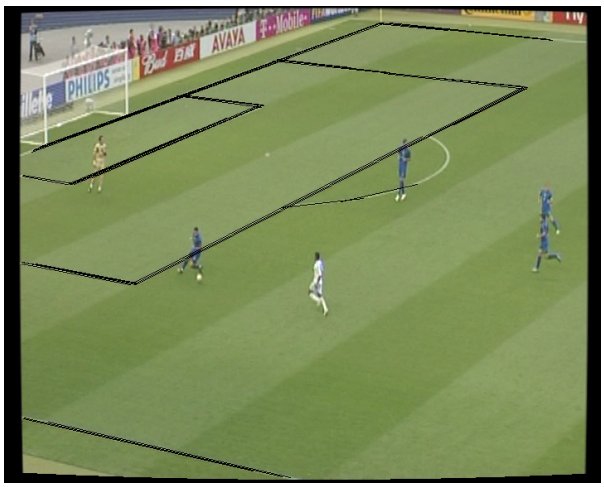


Figure 5.26 Extracted line segments by the Hough transformation (36 line segments).

However, the width of a field line in the image is typically between 1 and 35 pixels and therefore the Hough transformation detects multiple line segments for each field line.

Image lines that are close to each other and that are likely to belong to the same model line are merged into a single image line. The distance of two line segments is determined by their distance in the parameter space of the Hough transformation, as well as by their endpoints.

Two lines are assumed to belong to the same model line if the following constraints are met.

- The angle between both lines is less than $2°$.
- The difference in their distances to the origin is less than 15 pixels.
- The distance between both line segments is less than 30 pixels.

The former two constraints ensure that both lines are almost collinear, i.e. both lines lie almost on the each other's medial axis. The latter constraint is required to ignore collinear line segments which are far from each other.

For efficiency reasons, we order the line segment array according to the angle to the x-axis of the image. The search for almost parallel line segments to a given line segment then reduces to a bidirectional search in the array and is stopped if the maximum angle difference is exceeded. In doing so, the number of pairwise line segment comparisons is reduced.

The minimum distance of a line segment $\boldsymbol{l}_1$ to a line segment $\boldsymbol{l}_2$ is the smallest distance between a point $\boldsymbol{p}_1$ lying on line $\boldsymbol{l}_1$ and another point $\boldsymbol{p}_2$ lying on line $\boldsymbol{l}_2$. Fortunately, one of these two points has to be an endpoint of one of the line segments, and the smallest distance can be found by only considering all endpoints. The smallest distance of an endpoint to a line segment is the minimum distance to its projection on the other line and the distances to the

endpoints of the other line segment. If the projection of an endpoint does not lie within the endpoints of the other line segment, the calculated distance is not considered.

Figure 5.27 shows two examples. Line segments and their medial axes are printed as black lines and thin dashed lines respectively. Blue lines indicate valid and red lines denote invalid distance measures, whereas the green lines show the minimum distance between the line segments.

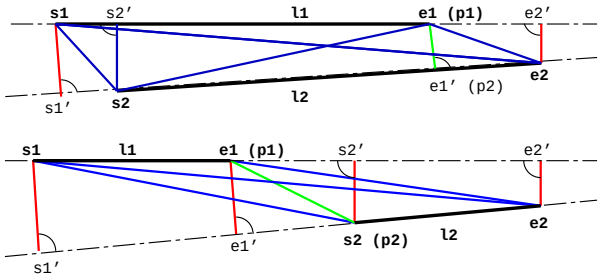


Figure 5.27 Line distance

After finding matching line segments, these are used to determine a new line segment by weighted least squares estimation. For this, we only consider the endpoints of all initial line segments weighted by the corresponding line segment lengths. These are then used to determine a new line by minimizing the weighted sum of squared distances between the line and the endpoints.

Using the homogeneous representation, the line is estimated as the least squares solution of the following equation.

$$A\boldsymbol{l} = 0 \tag{5.55}$$

with

$$A = (\boldsymbol{s}_1, \boldsymbol{e}_1, \ldots, \boldsymbol{s}_n, \boldsymbol{e}_n)^T$$

where $\boldsymbol{l}$ is the homogeneous representation of the new line and the rows of A contain the weighted coordinates of the endpoints of all matching (n) initial line segments.

As can be seen in equation (5.55), the homogeneous line $\boldsymbol{l}$ is defined up to a non-zero scale factor λ and the equation system can be solved with the OLS method by setting the last component of $\boldsymbol{l}$ to 1 (see section 4.1.1). As the last component of a homogeneous line is proportional to its distance to the origin, above method would fail for all lines going through the origin and we therefore use the eigenvalue decomposition to determine the new line. The solution of above equation is given by the unit eigenvector of A^TA corresponding to the smallest eigenvalue [HZ03, PTVF92]. Alternatively, one can use the *Singular Value Decomposition* of A.

Then, the solution is given by the unit singular vector corresponding to the smallest singular value of A [HZ03, PTVF92].

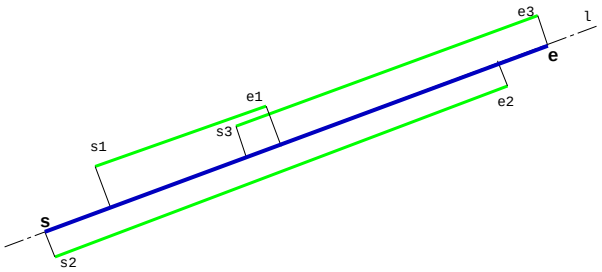


Figure 5.28 Merging line segments: Determining the endpoints of the new line segment.

After the new line is found, the endpoints of the new line segment are determined by projecting the endpoints of all initial line segments to the new line and selecting the two points with the largest distance to each other (see figure 5.28). This is done efficiently by selecting an arbitrary projected point, and calculating the signed distances to all projected endpoints (including the selected point). The signed distance d_{pq} of point $\boldsymbol{p}$ to point $\boldsymbol{q}$ is given by

$$d_{pq} = \begin{cases} \frac{q_x - p_x}{v_x} & \text{for} \quad v_x \neq 0 \\ \frac{q_y - p_y}{v_y} & \text{for} \quad v_x = 0 \end{cases} \tag{5.56}$$

where $\boldsymbol{v}$ is the normalized direction of the new line $\boldsymbol{l}$. The points with minimum and maximum signed distance to the selected point are used as the endpoints of the new line segment.

Figure 5.29 shows the resulting 11 line segments, which are merged from initially 36 line segments from figure 5.26.

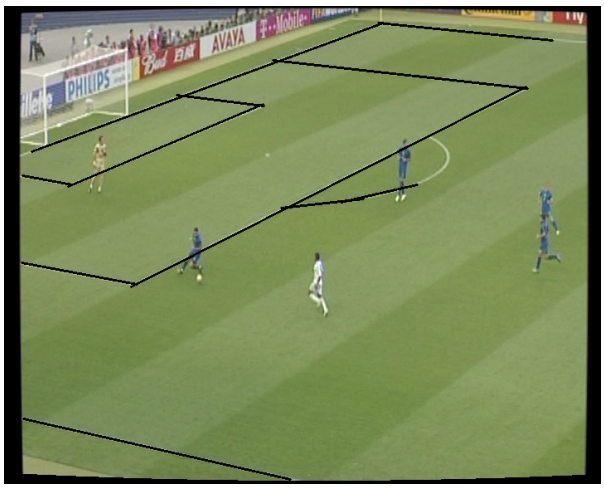


Figure 5.29 Merged line segments

5.5.1.3 Grouping of Parallel Line Segments

For further processing it is important to reduce the number of line segments, and to group line segments which belong to parallel field markings.

However, the mapping of model lines to image lines is a general projective transformation. And as a general projective transformation does not preserve parallelism, the angle between two image lines is not a good indicator for corresponding parallel model lines.

We determine image lines corresponding to parallel model lines by the following consideration. In projective plane $\mathbb{P}^2$, all parallel lines meet in a single ideal point, the *vanishing point* and all ideal points lie on the line at infinity $\boldsymbol{l}_\infty$

$$\boldsymbol{l}_\infty \simeq \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5.57}$$

A general projective transformation can map ideal points to finite points and vice versa. If ideal points are transformed into finite points, these points lie on the *horizon* or *vanishing line*[8].

If the vanishing points lie at infinity, the vanishing line stays at infinity, thus no (finite) horizon exists. This special case does not occur in the camera configuration that we consider, as the camera would have to point perpendicular towards the field plane from either above or below.

Nevertheless, one of the vanishing points can lie at infinity, in the case that the camera is directed perpendicular to the horizontal or vertical field lines. But as the roll angle of the cameras are usually close to 0, the direction of such an ideal point is then almost parallel to the x-axis of the image[9].

In the context of soccer broadcasts, the following constraints can be derived for the vanishing points.

- All image lines, which belong to the straight field markings, go through either one of the two vanishing points.
- At least one of these vanishing points is finite.

[8] a projective transformation preserves linearity

[9] This special case must not be handled in a different way, as the presented method uses points and lines in projective space

- Finite vanishing points have negative image y-coordinates, as the cameras are mounted above the field and the horizon is not visible.

- The direction of infinite vanishing points are almost parallel to the x-axis of the image.

It can be concluded from the above constrains that the vanishing line must be finite and almost horizontal, but beyond the image border.

We need to determine the two vanishing points meeting above constraints that have the highest support among the detected line segments.

Determining Vanishing Points

The determination of vanishing points is two staged. In the first stage we determine the intersection points of each two lines. All finite intersection points with positive y values and all infinite intersection points where the angles between line directions and image x-axis exceeds 10° are rejected. Otherwise, the number of supporting line segments (inliers) is determined for each intersection point. Intersection points with at least 3 inliers are refined by a weighted least squares using all inliers and are used as potential vanishing points.

Due to the duality of points and lines in projective plane $\mathbb{P}^2$, equation (5.55) is used to determine the least squares solution by interchanging lines and points [HZ03], where the entries of A contain the line parameters which are scaled to the line segments lengths.

In the next stage, we iterate over all combinations of two vanishing points, and determine the corresponding vanishing line. If the angle between the vanishing line and the x-axis of the image is greater than 10°, the combination is rejected, otherwise the combination is accepted. Within all accepted combinations, we select the combination with the most inliers where the sum of inliers of both vanishing points is maximal.

Finally all lines are assigned to two groups given by the obtained vanishing points. Lines, which do not either belong to one of the two groups are marked as outlier and are not considered in the further processing.

Determining Inliers

Inlying line segments for a calculated intersection point are determined by the maximum angle of the line segment to the direction vectors from the intersection point to their endpoints. If the maximum angle is below a given threshold (7°), the line is directed towards the intersection point, and is marked as an inlier. For intersection points lying at infinity, the direction vector is given by its first two components. For general case, the direction vector $\boldsymbol{d}$ defined by the

intersection point $\boldsymbol{p}$ and one of the endpoints $\boldsymbol{s}$ of the line $\boldsymbol{l}$ can be determined as follows.

$$\boldsymbol{d} = \begin{pmatrix} s_x \cdot p_z - p_x \\ s_y \cdot p_z - p_y \\ 0 \end{pmatrix} \tag{5.58}$$

Figure 5.30 shows the intersection point $\boldsymbol{p}_{1,2}$ of the lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. The angles between $\boldsymbol{l}_3$ and the direction vectors defined by the endpoints $\boldsymbol{s}, \boldsymbol{e}$ and the intersection point $\boldsymbol{p}_{1,2}$ are given by δ_1 and δ_2 respectively. The line segment $\boldsymbol{l}_3$ is tested for support of the calculated intersection point by looking at the angles δ_2 and δ_1 between the direction vectors defined by the endpoints $\boldsymbol{s}$, $\boldsymbol{e}$ and the intersection point $\boldsymbol{p}_{1,2}$. As the maximal angle δ_1 in this example exceeds the threshold, the line segment $\boldsymbol{l}_3$ is treated as an outlier.

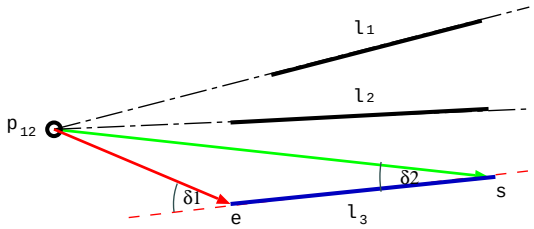


Figure 5.30 Determining vanishing points from line segments

Figure 5.31 shows grouped image lines. The lines that are deemed to belong to horizontal and vertical model lines are printed in red and blue respectively. Black lines show image lines which are neither assigned to horizontal nor to vertical field lines. The vanishing points

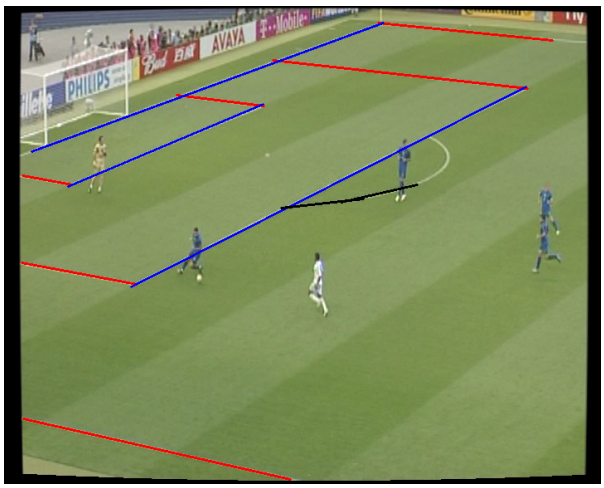


Figure 5.31 Grouped image lines

for the red and blue image lines are both finite and were estimated as the image coordinates $(-4151.69, -439.94, 1)^T$ and $(1789.62, -464.23, 1)^T$ respectively for the given example in figure 5.31.

5.5.2 Estimating Initial Parameters from Field Lines

As mentioned above, initial estimates for the focal length and the pan and tilt orientation in a new scene can be determined from extracted image lines without the need to identify the corresponding model lines. In fact, we determine these parameters from the vanishing points extracted from the image lines as described in section 5.5.1.

Due to the fact, that lines were extracted from rectified images (see section 5.5.1.1), we do not need to consider lens distortions in the following calculations any more. Furthermore, the lines and thus the vanishing points are transformed beforehand by the inverse of K_a and the camera matrix reduces to K_p (see equation (3.13)).

5.5.2.1 Determining the Focal Length

Let $\boldsymbol{p}$ and $\boldsymbol{q}$ be both vanishing points for horizontal and vertical model lines, and therefore the projections of the two ideal points $(\pm 1, 0, 0)^T$ and $(0, \pm 1, 0)^T$. As the vectors from the camera center to the ideal points are perpendicular, we can use the method described in section 5.3.1.2 to determine the focal length.

Equation (5.36) can be rewritten to

$$\begin{aligned} \frac{\boldsymbol{p}^T \omega \boldsymbol{q}}{\sqrt{\boldsymbol{p}^T \omega \boldsymbol{p} \boldsymbol{q}^T \omega \boldsymbol{q}}} &= 0 \\ \boldsymbol{p}^T \omega \boldsymbol{q} &= 0 \end{aligned} \tag{5.59}$$

with ω being the *Image of the Absolute Conic* (IAC) (see equation (5.24)). The focal length can be determined as:

$$f = \sqrt{-\frac{p_x \cdot q_x + p_y \cdot q_y}{p_z \cdot q_z}} \tag{5.60}$$

It can be seen in the denominator of the right hand side, that the focal length can only be determined if both vanishing points are finite.

5.5.2.2 Determining the Tilt Angle

Let $\boldsymbol{n}_w = (0, 0, 1, 0)^T$ be the normal of the soccer field in world coordinate system (WCS). The transformed normal $\boldsymbol{n}_c$ in camera coordinate system only depends on the rotation matrix $\breve{R}_c$ of the camera, and is given by

$$\boldsymbol{n}_c = \tilde{R}t \cdot \boldsymbol{n}_w \tag{5.61}$$

which is the last column of $\breve{R}_c$ (see equation (3.11)).

However, $\boldsymbol{n}_c$ can also be determined using the horizon of the soccer field in the image. The horizon or vanishing line $\boldsymbol{l}$ is specified given the two vanishing points $\boldsymbol{p}$ and $\boldsymbol{q}$ and can be determined by their cross product.

$$\boldsymbol{l} = \boldsymbol{p} \times \boldsymbol{q} \tag{5.62}$$

The normal $\boldsymbol{n}_c$ is then given by:

$$K^T \boldsymbol{l} = \boldsymbol{n}_c \tag{5.63}$$

Setting equations (5.61) and (5.63) to equal results in

$$\begin{pmatrix} f \cdot l_x \\ f \cdot l_y \\ l_z \end{pmatrix} = \begin{pmatrix} \sin(\beta) \\ -\sin(\alpha) \cdot \cos(\beta) \\ \cos(\alpha) \cdot \cos(\beta) \end{pmatrix} \tag{5.64}$$

and the tilt angle α can be obtained from the last two rows:

$$\alpha = \operatorname{atan2}(-f \cdot l_y, l_z) \tag{5.65}$$

Since the roll angle β is close to 0, the cosine of the roll angle is positive and the tilt angle can be determined uniquely.

5.5.2.3 Determining the Pan Angle

At this point, the focal length f and the tilt angle α have already been determined. The only remaining unknown parameter, the pan angle γ, depends on the assignment of the vanishing points to the ideal points $(\pm 1, 0, 0)^T$ and $(0, \pm 1, 0)^T$. Therefore it can not be determined

uniquely without additional information.

As both vanishing points correspond to perpendicular field lines, one of the points is redundant and does not provide additional information. This means that all pan angles that are consistent with one of the vanishing points, are automatically consistent with the other vanishing point too. The selected vanishing point can be assigned to both ideal points, and as the sign of the ideal points are unknown, four solutions exists.

Since the camera position is known, it can be used to eliminate two of the false solutions, as for these solutions the field lies behind the camera. By additionally supporting a valid pan range for the specific camera, the last wrong solution can be removed in most cases. Fortunately, the valid range for the pan angle can be obtained automatically by the camera position as well. Thus, the camera position is sufficient to determine the pan angle uniquely in most cases.

From the camera position, the sign of at least one vanishing point can be obtained. For example, for the main camera placed on the main stands, it is obvious that the vanishing point for vertical model lines corresponds to the ideal point in positive direction $(0, 1, 0)^T$. Whereas for a camera placed behind the right goal (e.g. the tactical camera with positive y-coordinate), the vanishing point for the horizontal model lines corresponds to the ideal point $(-1, 0, 0)^T$. A special case occurs for cameras placed "behind" a corner. In this case, the signs of both ideal points can be fixed. With this consideration, we select an ideal point with known sign and assign it alternately to both vanishing points. From each assignment, we obtain one solution for the pan angle and accept the one which lies within the valid pan range, as the other one is shifted by $90°$.

Suppose, the ideal point for the horizontal model lines is given by $(\varsigma, 0, 0)^T$, where $\varsigma \in \{+1, -1\}$ is the known direction of the ideal point. Both solutions for the pan angle γ can then be obtained by the following two equations

$$\boldsymbol{p} \simeq K\breve{R}_c \cdot \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \qquad \boldsymbol{q} \simeq K\breve{R}_c \cdot \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \tag{5.66}$$

Note, that the translation has no effect on ideal points, as they have a homogeneous coordinate of 0 and are independent with respect to the position of the camera.

We will now show how to derive the pan angle from the vanishing point $\boldsymbol{p}$. The second solution is similarly obtained by interchanging $\boldsymbol{p}$ by $\boldsymbol{q}$.

Thanks to the orthonormality property of the rotation matrix, we can eliminate the unknown scale factor in equation (5.66) by first multiplying with the inverse camera matrix, and then normalizing left hand side.

$$\tilde{\boldsymbol{p}} = \frac{K^{-1}\boldsymbol{p}}{\|K^{-1}\boldsymbol{p}\|} = \breve{R}_c \cdot \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \tag{5.67}$$

By decomposing the rotation matrix into three basis rotations (see section 2.2.1), we get:

$$\tilde{\boldsymbol{p}} = R_x R_y R_z \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \tag{5.68}$$

Now we can multiply with the inverse of the already known basis rotations and get:

$$R_z^T \underbrace{R_y^T R_x^T \tilde{\boldsymbol{p}}}_{=\boldsymbol{p}'} = \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \tag{5.69}$$

which is explicitly given by following equation:

$$\begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} \varsigma \\ 0 \\ 0 \end{pmatrix} \tag{5.70}$$

It can be seen that $\boldsymbol{p}'$ has to be perpendicular to the second row of the rotation matrix. By additionally considering the sign of the ideal point, the solution is obtained as:

$$\gamma_1 = \operatorname{atan2}\left(\varsigma p'_y, \varsigma p'_x\right) \tag{5.71}$$

As mentioned above, the second solution for the pan angle is similarly obtained by replacing

$\boldsymbol{p}$ with $\boldsymbol{q}$

$$\gamma_2 = \operatorname{atan2}\left(\varsigma q'_y, \varsigma q'_x\right) \tag{5.72}$$

The last row of Equation (5.70) implies that $\boldsymbol{p}'_z$ has to be 0. That means that we can obtain an estimate for the accuracy of the determined focal length and tilt angle by looking at the resulting z value. In fact, this shows how good the vanishing points could be determined.

The method above requires that the sign of the horizontal ideal point could be fixed. However, if the sign of the ideal point corresponding to the vertical model lines is known, equation (5.70) becomes:

$$\begin{pmatrix} \cos(\gamma) & \sin(\gamma) & 0 \\ -\sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{pmatrix} 0 \\ \varsigma \\ 0 \end{pmatrix} \tag{5.73}$$

and the two solutions are given by:

$$\gamma_1 = \operatorname{atan2}\left(-\varsigma p'_x, \varsigma p'_y\right)$$
$$\gamma_2 = \operatorname{atan2}\left(-\varsigma q'_x, \varsigma q'_y\right)$$

To eliminate the wrong solution, we determine the valid pan range of the specific camera using the knowledge of the field size and the camera position. TV cameras are usually placed $30m$-$50m$ from the next field line, so that the field is only visible in a small range of the pan angle. As the wrong solution is shifted by $90°$, it can be eliminated if the valid range is smaller than $90°$. In TV broadcasts, the valid range is usually smaller than $70°$, so that the pan angle can be identified uniquely.

The valid pan range is determined by the camera position and the field size as the range between the minimum and maximum angle, where at least one corner point is visible in the horizontal image center.

5.5.3 Validating the Estimated Parameters

At this stage, the parameter set (pan, tilt, zoom) has been determined without the need to identify the image lines. However, due to noise and the fact that the video segment may be captured by a camera other than from the camera of interest, the determined parameters may

be out of the convergence range of the tracker or even completely wrong.

For this reason, it is required to confirm or reject the parameters by comparing the expected projections of the model lines with the extracted image lines. For each projected model line, the nearest neighbor is determined and is counted as an outlier, if this distance is beyond a given threshold. If more than one outlier is detected, the determined parameters are rejected, and the next image in the video sequence with sufficient image lines is processed. On the other hand, if the projected model lines are consistent with the image lines, the estimated parameters are provided to the Iterative Camera Calibration module (ICC) as the initial estimate.

The distance of a projected model line to an image line is determined as described in section 5.5.1.2 except for the threshold values. The threshold value for the line orientation is chosen to be $5°$, the maximal distance difference to the origin by 30 pixels, and the maximum line distance is selected to be 30 pixels. Additionally, we determine the ratio of overlap between image and model lines by endpoint projection.

The above constraints ensure, that the chosen parameters are within the convergence range of the tracker.

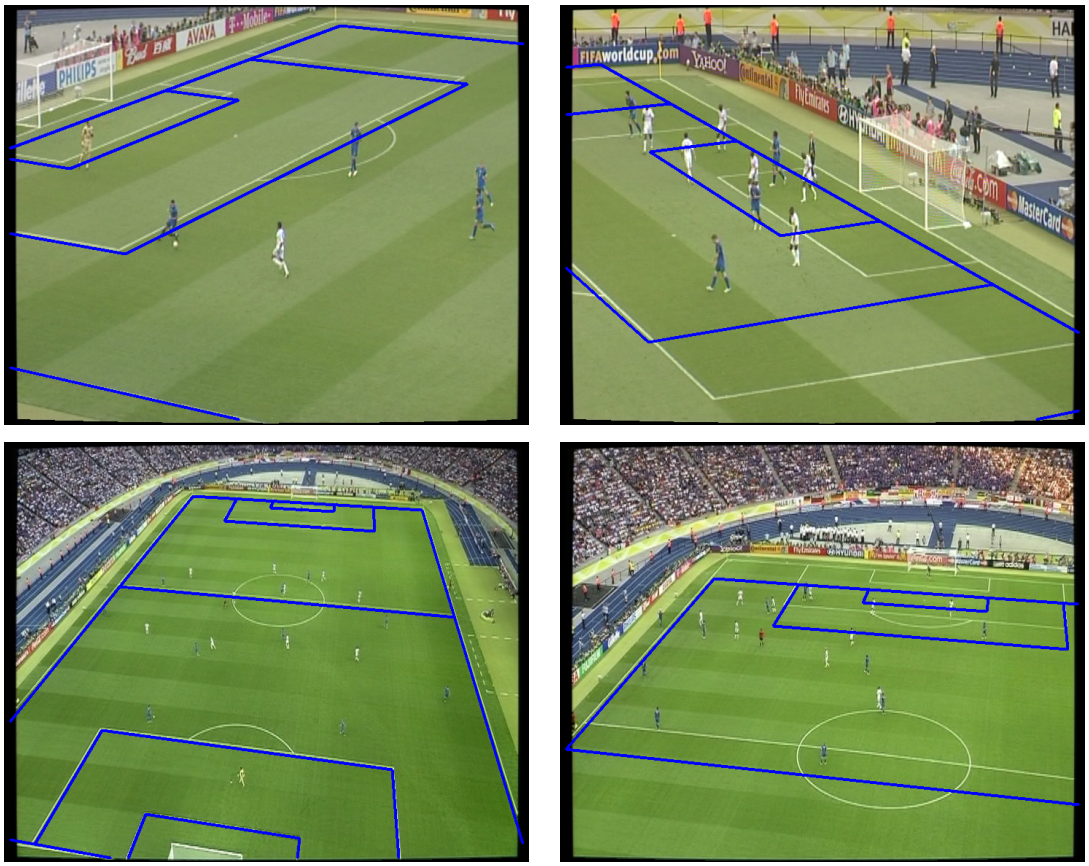


Figure 5.32 Results for the Initial Parameters Estimation: determined parameters are within (left) or out of (right) the convergence range of the tracker (ICC).

Figure 5.32 depicts the results for two different camera configurations with two selected images each. The field model is projected into the image to show the accuracy of the described method. The results for the example given by figures 5.24-5.29 and 5.31 is shown in the top left image. Parameters, that are out of the convergence range of the tracker are depicted in the second column. In these cases other frames can be selected for the initial parameter estimation, or they can be estimated by human intervention.

5.5.4 Covariance Matrix of the Initial Estimate

If valid parameters could be determined, then these are composed to the state vector $\tilde{\boldsymbol{x}}_0$ as follows:

$$\tilde{\boldsymbol{x}}_0 = \begin{pmatrix} f \\ \alpha \\ \gamma \end{pmatrix} \tag{5.74}$$

Since the focal length is given in millimeters and the angles in radians, the covariance matrix of the initial estimate is selected as:

$$\tilde{P}_0 = \begin{pmatrix} 3.0 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{pmatrix} \tag{5.75}$$

From the diagonal structure of the covariance matrix, it can be seen that the parameters are supposed to be uncorrelated, and from the large diagonal elements it can be seen that the state vector is supposed to be very uncertain. The reason for this is, that a sufficient number of field lines are visible and the initial guess is within the convergence rate of the tracker. This means, that for the first frame the MAP estimation can converge in almost all cases towards the true system state without the need of the stabilizing effect of the predicted parameters (initial guess). Unlike the predicted parameters, the initial guess is imprecise, and therefore is only required to obtain line correspondences and as the initial estimate for the nonlinear optimization (see next section). In fact, using a smaller covariance matrix would lead to a worse final estimate for the first frame.

5.6 Iterative Camera Calibration

The Iterative Camera Calibration module (ICC) successively estimates the dynamic parameters for a video sequence until a shot cut is detected.

The ICC is integrated into the Kalman Filter (KF) framework and uses a modified version of the weighted Gauss-Newton approach with prior knowledge to robustly obtain the MAP estimation in the measurement update stage. The KF is commonly used for solving parameter estimation problems in the computer vision community [Fau93] mostly because it is the linear, recursive and *optimal* filter (see chapter 4).

The tracking task is depicted in figure 5.33.

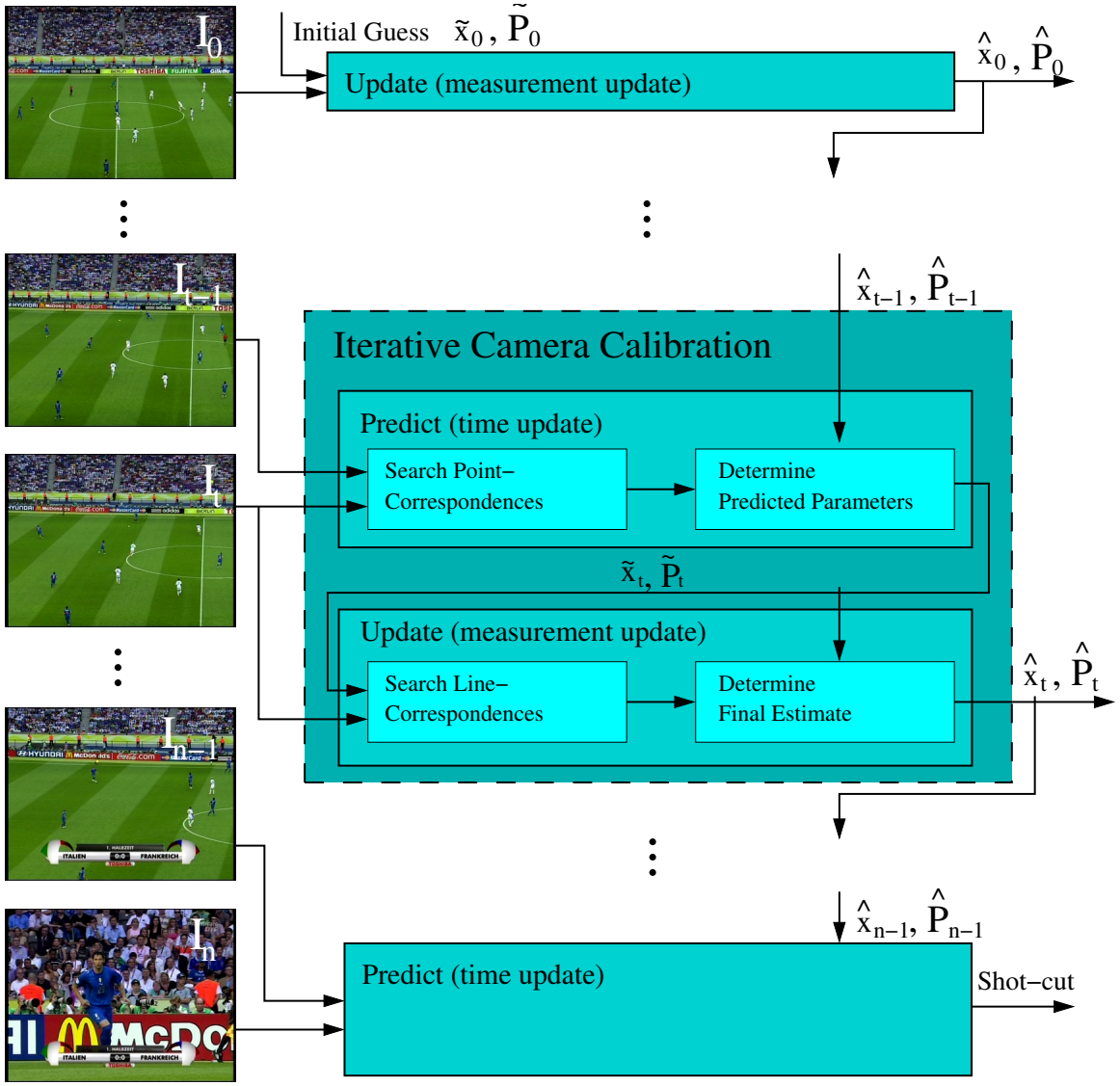


Figure 5.33 The *Iterative Camera Calibration* module

In each time step, an initial guess including uncertainties for the parameters to be estimated is required. At the beginning of a sequence, this *a priori* information is usually provided by the user or is obtained from the *Initial Parameter Estimation* module as described in section 5.5. During the tracking task, however, parameters that have been obtained by the optical flow information between two subsequent images are used for prediction. If the optical flow is inconsistent with the expected camera motion (pan, tilt and zoom), a shot cut is assumed and the ICC returns the control to the *Scene Iterator*.

In the update stage, correspondences between the model and image lines are sought in the vicinity of their expected positions. The expected positions for the image lines are given by the projections of the corresponding model lines using the initial guess. To speed up correspondence search, we use line-point correspondences instead of line-line correspondences. This means, that we search for correspondences between the several selected points on each model line and their matching points in the image along the projected line normals in both directions. Compared to the extraction of image lines, this method reduces the effort to determine the expected color transitions along several lines. Furthermore, we restrict the search of corresponding image points to the vicinity of the projected model line/point. The vicinity of a model line/point is defined by a search width, which is selected to be at least the confidence interval of 99.7%. The interval value is obtained from the uncertainty of the initial guess. The final estimate is then robustly determined by minimizing the distances of the image points to the projected model lines considering the initial parameters. The used cost function, e_{ln}, is described in section 5.2.2.

Unfortunately, using image points instead of image lines leads to a loss in accuracy and robustness, as the image lines are very robust features. To compensate for the so-caused limited accuracy, multiple image points for each model line are used. The required robustness is achieved by integrating the robust M-Estimators (see section 4.2.5.4) into the weighted Gauss-Newton method (see section 4.2.3.1), which we call the *reweighted* Gauss-Newton method for the remainder.

The robustness and accuracy of the update stage and thus of the whole tracker highly depends on the quality of the initial guess. If the initial parameters are substantially far from the true state, the number of correct line correspondences found is low while the number of outliers increases, which makes the accurate estimation of the parameters difficult. In such a case, the monitoring process running during the update stage invokes the *Initial Parameter Estimation* to reinitialize the tracking task. If also this fails, the control is returned to the *Scene Iterator*, which searches for the next suitable sequence. The range for the initial parameters leading to a valid estimate is called the *convergence range* of the tracker.

5.6.1 Parameter Prediction

The time update stage of the Kalman Filter predicts the state vector for the next time step by using the motion model (see equation (4.2)). In the camera tracking task, the motion model can be simplified by only adding an input control vector $\boldsymbol{u}_t$ to the last estimate $\hat{\boldsymbol{x}}_{t-1}$.

$$\tilde{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t-1} + \boldsymbol{u}_t \tag{5.76}$$

The control vector $\boldsymbol{u}_t$ governs the relative motion of the camera from time $t-1$ to time t. Above representation suits to the motion model given in chapter 4. But instead of obtaining the control vector, we directly estimate the predicted state using the reweighted Gauss-Newton method.

This is efficiently done using optical flow between two subsequent images. 100 randomly selected points around the image border are tracked between both images as described in section 5.4.1.2. The processed points are depicted as blue dots in figure 5.34.

As the considered camera configurations suppose fixed camera centers, the transformation between two images taken by the same camera with different orientation and/or focal lengths can be described by a projective transformation. Let $\boldsymbol{w}$ be a point in the world coordinate system, $\boldsymbol{q}_{t-1}$ its projection in image I_{t-1} and $\boldsymbol{p}_t$ the projection of the same point in image I_t. Furthermore, let K_{t-1}, R_{t-1}, K_t and R_t be the corresponding camera and rotation matrices for both images. The projection of both points can then be written as:

$$\boldsymbol{q}_{t-1} \simeq K_{t-1} R_{t-1} T \boldsymbol{w} \tag{5.77}$$

$$\boldsymbol{p}_t \simeq K_t R_t T \boldsymbol{w} \tag{5.78}$$

As the camera position is supposed to be fixed, the translation given by T is the same for both views. Thereby we can transform $\boldsymbol{w}$ by T beforehand. Let $\boldsymbol{w}_c = T\boldsymbol{w}$, then $\boldsymbol{w}_c$ can be determined (up to scale) by equation (5.77) as follows:

$$\boldsymbol{w}_c \simeq R_{t-1}^T K_{t-1}^{-1} \boldsymbol{q}_{t-1} \tag{5.79}$$

Integrating this result into equation (5.78) yields:

$$\boldsymbol{p}_t \simeq \underbrace{K_t R_t R_{t-1}^T K_{t-1}^{-1}}_{H} \boldsymbol{q}_{t-1} \tag{5.80}$$

The 3×3 matrix H is called Homography, and is defined up to a non-zero scale factor λ. Thus, H has 8 degrees of freedom and can be determined using at least four known and non-collinear point correspondences (see section 5.3.1).

Point correspondences are found by using the method described in section 5.4.1.2 and transformed by the inverse of K_a beforehand. Both camera matrices reduce to the form of K_p (see equation (3.13)) with their corresponding focal lengths f_{t-1} and f_t.

The homography is robustly estimated using RANSAC [HZ03] and refined by a least squares solution using all inliers.

5.6.1.1 Calculating the Predicted Parameters

The predicted parameters can be determined by two different approaches. One is the closed form solution, and is obtained by using the above representation of the homography. The other method uses a nonlinear least squares estimation that makes it possible to consider several constraints (e.g. constant roll angle). In the context of this thesis, the least squares estimation is used, but we will present both methods, as the closed form solution is used for the initial estimate for the least squares solution.

Closed Form Solution

The closed form solution is obtained using the above representation of the Homography (see equation (5.80)). Since the camera and rotation matrices for the previous time step are already known, equation (5.80) can be rewritten to:

$$\underbrace{R_{t-1}K_{t-1}H}_{=H'} = \lambda K_t R_t \tag{5.81}$$

or in the row representation:

$$\begin{pmatrix} \boldsymbol{h'}_1^T \\ \boldsymbol{h'}_2^T \\ \boldsymbol{h'}_3^T \end{pmatrix} = \begin{pmatrix} \lambda f_t \boldsymbol{r}_1^T \\ \lambda f_t \boldsymbol{r}_2^T \\ \lambda \boldsymbol{r}_3^T \end{pmatrix} \tag{5.82}$$

with $\boldsymbol{h'}_1^T$, $\boldsymbol{h'}_1^T$ and $\boldsymbol{h'}_1^T$ being the rows of matrix H', and $\boldsymbol{r}_1^T$, $\boldsymbol{r}_1^T$ and $\boldsymbol{r}_1^T$ being the rows of the rotation matrix R_t. From the orthonormality property of the rotation matrix, the unknown

scale factor λ is obtained as the length of the last row of matrix H'.

$$\lambda = \sqrt{\boldsymbol{h'}_3^T \boldsymbol{h'}_3}$$
$$H'' = \frac{1}{\lambda} H' \tag{5.83}$$

Then the focal length can be obtained as the length of one of the first rows of matrix H''.

$$f_1 = \sqrt{\boldsymbol{h''}_1^T \boldsymbol{h''}_1}$$
$$f_2 = \sqrt{\boldsymbol{h''}_2^T \boldsymbol{h''}_2}$$

Due to noise, both lengths may be different, and so we use the geometric mean of both to obtain the final focal length.

$$f_t = \sqrt{f_1 \cdot f_2} \tag{5.84}$$

After the camera matrix is determined, the rotation matrix can be obtained by

$$K_t^{-1} H'' = R_t \tag{5.85}$$

Since it is not assured that the obtained rotation matrix is orthonormal, the final rotation matrix is determined by the best orthonormal matrix in terms of the smallest Frobenius norm [Zha00]. The pan and tilt angles are then extracted from the obtained rotation matrix as denoted in equation (2.7).

Least Squares Solution

The least squares estimate uses the reweighted Gauss-Newton method to determine the final parameters. The results from the closed form solution presented in previous section, are used in several ways. First, the closed form solution is used as the required initial guess for the nonlinear least squares method. Secondly, all outliers obtained by RANSAC from previous step are eliminated beforehand. Finally, the determined homography is used to estimate the variances for the point correspondences.

The covariance matrices for the points $\boldsymbol{p}_i$ are assumed to be homoscedastic, i.e. $\Sigma_i = \sigma_i^2 I$, and the variances σ_i^2 are estimated by following heuristic. Let $\boldsymbol{q}_i$ be the point in image I_{t-1} and $\boldsymbol{p}_i$ their corresponding points in image I_t. Furthermore, let H be the inter image homography describing the transformation between two subsequent images from time $t-1$ to time t. The

variance is then given by the squared residual distances as follows:

$$\sigma_i^2 = (\boldsymbol{p}_i - H\boldsymbol{q}_i)^T(\boldsymbol{p}_i - H\boldsymbol{q}_i) \tag{5.86}$$

where the homogeneous points $\boldsymbol{p}_i$ and $H\boldsymbol{q}_i$ are normalized, so that the homogeneous component is equal to 1. The variance is simply the squared distance of the corresponding points in pixel coordinate system plus 1.

The cost function for the least squares solution can be inferred by following consideration. The direction of the ray from the camera center through the point $\boldsymbol{q}_i$ in world coordinate system is given by:

$$\boldsymbol{w}_i = R_{t-1}^T K_{t-1}^{-1} \boldsymbol{q}_i \tag{5.87}$$

By interpreting this direction as an ideal model point in world coordinate system, the cost function e_{pt} can be used (see section 5.2.1).

The resulting parameters and the corresponding covariance matrix are obtained similarly as described in section 5.6.3 using the reweighted Gauss-Newton method. The difference is that no prior knowledge is available, and that the measurement variance is obtained additionally.

Let $\boldsymbol{x}_{k'}$ be the estimated parameter vector and $\Sigma_{k'}$ its covariance matrix from the last iteration k'. Then, the predicted parameter vector and its covariance matrix are given by:

$$\begin{aligned} \tilde{\boldsymbol{x}}_t &= \boldsymbol{x}_{k'} \\ \tilde{P}_t &= \hat{P}_{t-1} + \Sigma_{k'} \end{aligned} \tag{5.88}$$

The unknown measurement variance $\hat{\sigma}^2$ is estimated from the last iteration and incorporated into $\Sigma_{k'}$. It is given by equation (4.100), where the covariance matrix of the measurements is replaced by the *corrected* covariance matrix Σ'_{zz} (see equation (5.95)).

5.6.1.2 Experimental Results

We have tested the parameter prediction method separately on several uncut video streams. The method was able to track the parameters for 1091 frames on average until the estimated parameters drifted out of the convergence range of the tracker. This means that the parameter prediction method could track the parameters without getting lost and without using the image lines for approximately 43 seconds [10] on average. The maximum time period for the prediction

[10] PAL: 25fps

without getting lost was 3303 frames (132 seconds [10]), which is more than two minutes of playing time.

Figure 5.34 shows the parameter prediction over 600 frames. It can be seen, that the predicted parameters for the last frame are still within the convergence range.

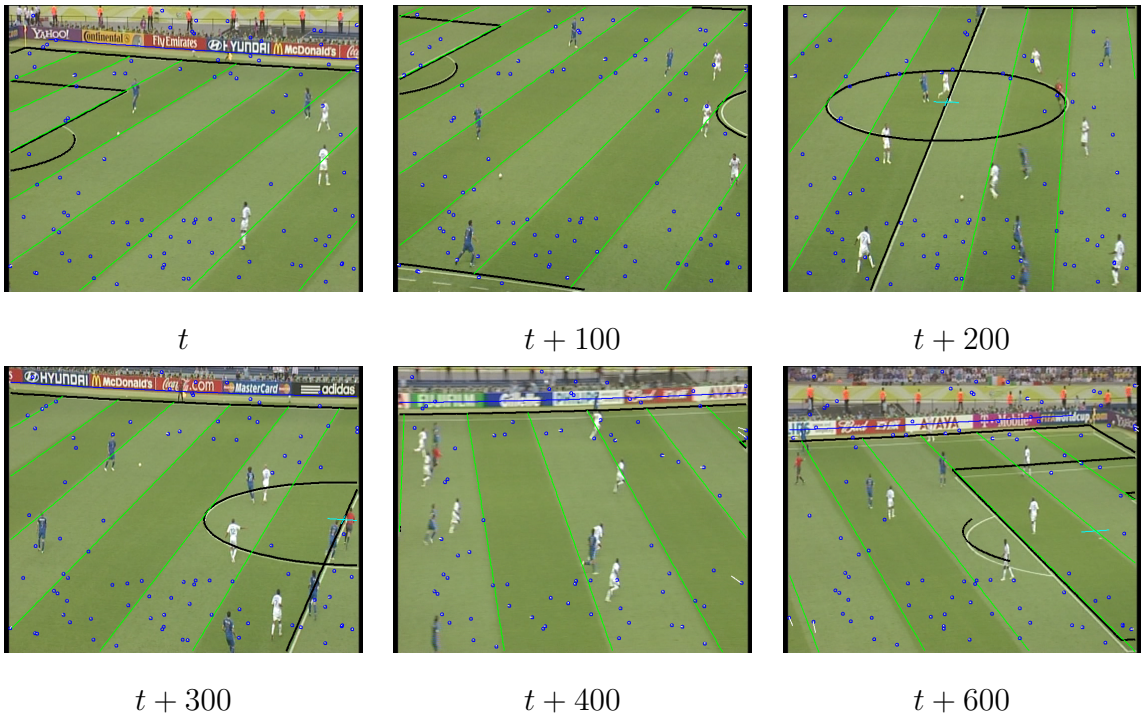


Figure 5.34 Parameter prediction over 600 frames. Established point correspondences are displayed as blue dots. The predicted parameters after 600 frames are still within the convergence range of the tracker.

For the scenes with zooming but only slow camera motion, the predicted parameters stay within the convergence range for more than 2200 frames (88 seconds [10]), whereas for scenes with zooming and very fast camera motion this rate goes down to only $10 - 20$ frames. The main reason for this is that in sequences with fast camera motion or zooming, the images are very blurry, and therefore the number of found point correspondences drops rapidly. Thus the parameters start drifting away from the true state.

However, as the prediction step is followed by an update step during tracking, the parameter prediction alone does not have to track the parameters for more than one frame.

5.6.2 Searching Line Correspondences

As mentioned before, only correspondences between image points and points lying on model lines are used during the update stage. The number of correspondences for a model line depends on its type (see section 5.1) and length. Field lines, which are the main features for the tracking task, provide $10 - 20$ correspondences each, depending on their length. For example, the center circle provides 20 correspondences, whereas the goal lines provide just 10 correspondences.

Field edges and lawn lines, are not very reliable and therefore provide only 7 and 5 correspondences respectively. They are mainly used to stabilize the scenes with a moderate number of visible field lines. In such cases, the correspondences from these lines prevent the parameters from drifting away until a sufficient number of field lines becomes visible again. On the other hand, if enough field lines are visible, the moderate amount of correspondences from field edges and lawn lines does not strongly influence the estimation results.

To ensure a uniform distribution of the selected model points along the visible line segment, a predefined sequence of 63 values for the line parameter s is tested until the required number of correspondences is found. A shortened parameter sequence with just 15 values for s is depicted in figure 5.35. If less than 3 correspondences for a single line could be found, the line

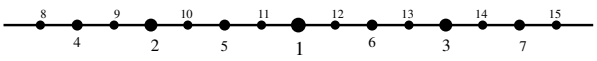


Figure 5.35 Order of the curve parameters for the correspondence search

is marked as invisible, which indicates that the line is either occluded by players or the initial parameters are out of the convergence range of the tracker. The latter case is important for the monitoring process, which is described in section 5.6.5.

The basic algorithm for all line types is described by the following steps.

- First, the model point $\boldsymbol{w}$ and its normal $\boldsymbol{n}$ for a given value for s are obtained from the corresponding curve (see section 5.1), and projected into the image using the predicted camera parameters. Additionally, the covariance matrix Σ_{qq} of the projected model point $\boldsymbol{q}$ is determined by using the linearization method as described in section 2.3.1.2.

- The search width ρ is selected as $3\sigma_n + 5d$, where σ_n^2 is the variance along the projected normal determined from Σ_{qq} as described in section 2.3.2. The variable d is selected to be $d = \max(d', 5)$ for all line types, where d' is the expected line width in pixels at the given model point.

- Finally, the corresponding image point (transition point) is sought along the normal in both directions within the search width. If a suitable point could be found, its variance is estimated and a correspondence D is built including the model point, found image point and its variance (see section 5.2).

The search lines for the model points for the example from figure 5.29 are depicted in figure 5.36. Since the transition along the projected line normal is sought, the variance along

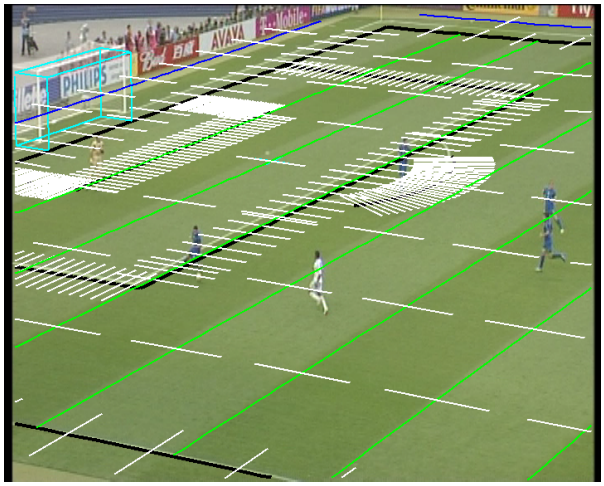


Figure 5.36 Search range for correspondence search

the search direction is estimated instead of the covariance matrix of the transition point. The main advantage is, that it is not required to propagate the variance of the transition point to the distance measure given by the used error function e_{ln} in the update stage (see section 5.2.2).

5.6.2.1 Searching Field Markings

Point correspondences for field lines are found using a one-dimensional template matching. The template describes the color transition for a field line, and is specified by the expected line width d', $d = \max(d', 5)$, and the color distributions (see figure 5.37).

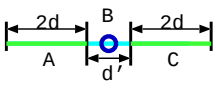


Figure 5.37 Template for line color transition

The color values within the segments A and C are supposed to belong to the field color, whereas the gray value in segment B is supposed to be considerably brighter than the gray

value of the segments A and C. The anchor for the template is in the center of the line (on the medial axis), and is depicted by the blue circle.

The template is shifted along the search range, and the match for each point (pixel under the anchor) is determined by a utility function. Points where the gray value of segment B is not considerably brighter than the gray value of the segments A and C, are rejected. The point with the best match is used as the corresponding image point, if the match is above a given threshold.

As the color within the segments A and C is supposed to be the field color, the utility function determines the number of pixels within both segments, that belong to the field color class. Instead of a binary assignment of pixels into field color and non-field color pixels, we assign a weight to each pixel representing the confidence that the pixel belongs to the field color. Therefore, we first determine the Mahalanobis distance to the field color class for each pixel color value within the segments A and C, and then sum up the values of a weight function taking the Mahalanobis distances. We use the weight function proposed in section 4.2.5.4 with the values 3 and 6 for the parameters a and b respectively. The threshold for the utility value is selected to be $2d$, since the perfect match has the utility value of $4d$.

The variance is heuristically determined as the distance to the closest point where the utility value is below 75% of the maximum. Figure 5.38 depicts the matching results for a model point. The black line shows the rejected utility values along the search range, whereas the red

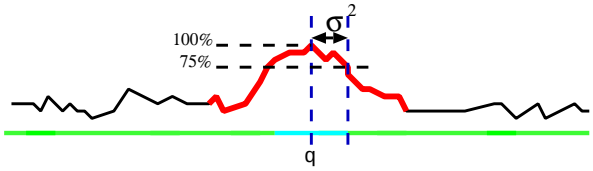


Figure 5.38 Template matching results for a line color transition

curve shows the accepted utility values. The dashed blue line shows the point with the best match, and the horizontal black lines show the peak level and the threshold for the variance estimation.

To decide, whether the segment B is brighter than segment A and C, we determine the sample means and sample variances of the gray values within these segments, and use the *Hotellings t-Test* [Leh86]. More specific we use the two sample t-Test, which decides whether the difference of the means of two distributions is larger than a given value g. In our case, g is the expected gray value difference between the field markings and the field, and has been set to 25 according to the experiences gathered during experimental evaluation.

Let μ_A, μ_B and μ_C be the sample means of the segments A, B and C respectively, and σ_A^2,

σ_B^2 and σ_C^2 their corresponding sample variances. We suppose, that the unknown real variances for all segments are caused by the same error sources and are therefore equal. The test values t_A and t_C are then given by:

$$t_A = \frac{\mu_B - \mu_A - g}{\sqrt{\frac{(2d-1)\sigma_A^2 + (d'-1)\sigma_B^2}{2d+d'-2}\left(\frac{1}{2d} + \frac{1}{d'}\right)}}$$
$$t_C = \frac{\mu_B - \mu_C - g}{\sqrt{\frac{(2d-1)\sigma_C^2 + (d'-1)\sigma_B^2}{2d+d'-2}\left(\frac{1}{2d} + \frac{1}{d'}\right)}} \tag{5.89}$$

As it is possible that the sample variance for a segment can be 0, we use a lower bound of 1 for all sample variances. For the case $d' = 1$, no sample variance for segment B can be obtained and we set its variance equal to the variance of segment A and C respectively. The degree of freedom (DoF) is $2d + d' - 2$ and unfortunately depends on the sample sizes. Instead of fixing the confidence level and obtaining the corresponding threshold $\bar{t}$, we use a fixed threshold of $\bar{t} = 2.55$. For the smallest line width (1 pixel), the confidence level is 97.5% and for the line width of 35 pixels for example, the confidence level is 99%[11]. The segment B is supposed to be considerably brighter than its neighbooring segments A and C, if both test values t_A and t_B are above the threshold $\bar{t}$.

5.6.2.2 Searching Lawn Lines

Lawn lines are purposely made by the lawn mower as additional auxiliary lines for the referees and are quite common in the soccer stadiums. Usually, they appear as pattern consisting of horizontal and vertical stripes, with alternating intensity values. Thus, lawn lines separate brighter regions from darker regions. The one-dimensional template for lawn lines is depicted in figure 5.39.

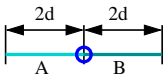


Figure 5.39 Template for lawn line color transition

The template is shifted along the search line, and for each point, we determine the sample mean and sample variance of the intensity values for both segments. Let μ_A and σ_A^2 be the mean and variance of the intensity values for the segment A, and μ_B and σ_B^2 for the segment B.

[11] The confidence levels can be obtained from the distribution tables.

To decide, whether one side is brighter or darker than the other side, we again use the *Hotellings t-Test*. Because of the fact, that the distributions are given by samples with equal size, the test criterion is given by:

$$t = \frac{\mu_A - \mu_B - g}{\sqrt{\frac{\sigma_A^2 + \sigma_B^2}{2d}}} \tag{5.90}$$

In case of lawn lines, the intensity difference g between dark and light stripes is set to 8, again based on our experiences from the experimental evaluation. The degree of freedom is $4d - 2 = 18$, since $d = 5$ is used. The threshold value for the confidence level of 99% is $\bar{t} = 2.55$, and is obtained from the distribution table. Since the brighter side of a lawn line is given by the field model, we suppose in the following that A belongs to the brighter side.

From all the points fulfilling the above criteria, we select the one with the largest test value t, since it is the point separating both regions most clearly. The variance for the best match is obtained from the following heuristics:

$$\sigma^2 = \frac{10.2}{t} + 1 \tag{5.91}$$

where t displays the test value for the best match. Therefore, the variance equals to 5 pixels in the worst case ($t = 2.55$) and 1 pixel in the best case. In practice, however, the variance mostly lies between 5 and 2 pixels.

5.6.2.3 Searching Field Borders

The field border along a search line is determined as the point, where the pixels on the one side belong to the field color, while those on the other side do not. The weights indicating the similarity between the pixel color and the field color are again obtained by the proposed weight function. The weight function takes the Mahalanobis distance of the pixel color to the field color class, and lies between 0 (no match) and 1 (perfect match).

The task is to find the boundary within the search line, where the sum of the weights on one side is large, while the sum of the weights on the other side is low. In the remainder, the method will be explained for a horizontal search line, where the field is supposed to be on the left side, i.e. we search for the transition of the right field border.

We iterate over the search line from left to right, and determine the sum of weights on the left side of each iterated point. The leftmost point that separates at least 98% of the total sum is supposed to be a border point. Figure 5.40 depicts this idea.

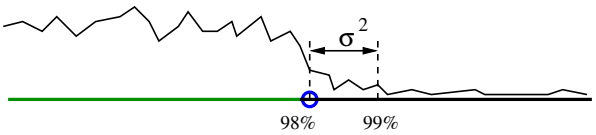


Figure 5.40 Finding field edges

The variance is again obtained through heuristics. If the border of the field is exact, the weights should give a rectangular profile within the search line and the variance should be small. On the other hand, if the border is blurry, the profile has a slope, and the variance should be larger. The variance is determined as the difference between the point at 98% and the point at 99%, but is set to at least 1.

Figure 5.41 shows the correspondence search results for all three line types for the example depicted in figure 5.36.

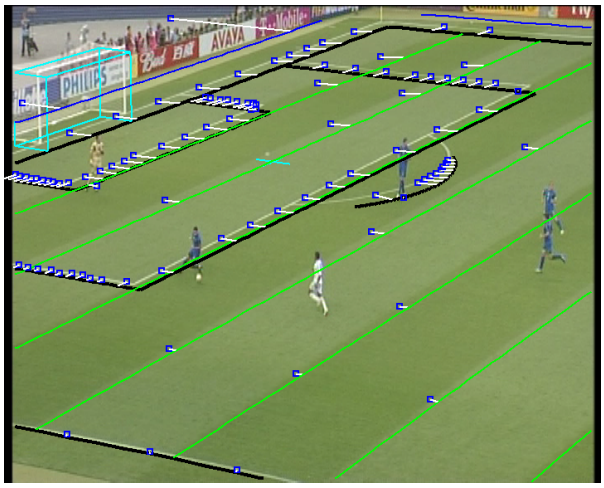


Figure 5.41 Line correspondences found by the methods described in section 5.6.2.

5.6.3 Parameter Update

The MAP estimation in the update stage is determined by the reweighted Gauss-Newton method with prior knowledge. The prior knowledge is given by the Parameter Prediction stage or obtained by the Initial Parameter Estimation module in the first step.

As mentioned earlier, the update stage uses line correspondences, that are found by using the predicted parameters as described in section 5.6.2. Each line correspondence D_i provides one error value r_i with corresponding variance σ_i^2. The error values are returned by the cost function e_{ln} as described in section 5.2.2. All error values and variances are stacked into the m-dimensional error vector (residual vector) $\boldsymbol{r}$ and the $m \times m$ covariance matrix Σ_{zz} respectively.

The covariance matrix Σ_{zz} has diagonal form, as we assume that the observations are made independently.

The algorithm of the reweighted Gauss-Newton is similar to algorithm 4.4. The main difference is, that the covariance matrix of the measurements is modified to suppress outliers. We will now present the reweighted Gauss-Newton method with prior knowledge for a single iteration.

Let $\boldsymbol{r}_k$ be the error vector at the current iteration k and the $m \times 3$ matrix J_k the Jacobian of $\boldsymbol{r}_k$. Furthermore, let $\tilde{\boldsymbol{x}}_t$ be the predicted parameters for the current time step t, and $\tilde{P}_t$ their covariance matrix. Note, $\boldsymbol{r}_k$ and J_k are obtained at the previous estimate $\boldsymbol{x}_{k-1}$ (see equation (4.101)).
According to equation (4.120), the correction step $\boldsymbol{l}_k$ for iteration k is obtained by:

$$J_k \boldsymbol{l}_k = -\boldsymbol{r}_k$$

Using equation (4.81) with $\rho_k = 1$, we can rewrite above equation as:

$$\begin{aligned} J_k \left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right) &= -\boldsymbol{r}_k \\ J_k \boldsymbol{x}_k &= J_k \boldsymbol{x}_{k-1} - \boldsymbol{r}_k \end{aligned} \tag{5.92}$$

Incorporating the predicted parameters into this equation leads to:

$$\begin{pmatrix} J_k \\ I_n \end{pmatrix} \boldsymbol{x}_k = \begin{pmatrix} J_k \boldsymbol{x}_{k-1} - \boldsymbol{r}_k \\ \tilde{\boldsymbol{x}}_t \end{pmatrix} \tag{5.93}$$

Assuming that the predicted parameters are independent from equation (5.92) leads the to following least squares estimate:

$$\begin{aligned} \boldsymbol{x}_k &= \left(J_k^T \Sigma_{zz}^{-1} J_k + \tilde{P}_t^{-1}\right)^{-1} \left(J_k^T \Sigma_{zz}^{-1} \left(J_k \boldsymbol{x}_{k-1} - \boldsymbol{r}_k\right) + \tilde{P}_t^{-1} \tilde{\boldsymbol{x}}_t\right) \\ \Sigma_k &= \left(J_k^T \Sigma_{zz}^{-1} J_k + \tilde{P}_t^{-1}\right)^{-1} \end{aligned} \tag{5.94}$$

Up to this point the only difference to equation (4.126) is that we now determine the corrected new state $\boldsymbol{x}_k$ instead of the correction vector $\boldsymbol{l}_k$ in each iteration.

To get a robust estimator, we intergrate the M-Estimators (see section 4.2.5.4) as follows. In each iteration, a diagonal weight matrix W is determined from the residuals of the last iteration. This weight matrix has the same function as the inverse covariance matrix in the

above equation (5.94). It is providing different weights to the measurements in order to give more confidence to some measurements than to others.

The inverse covariance matrix Σ_{zz}^{-1} in above equation is replaced by another weight matrix $\Sigma_{zz}'^{-1}$, which is determined in each iteration as follows:

$$\Sigma_{zz}'^{-1} = \Sigma_{zz}^{-1} W^{(k-1)} \tag{5.95}$$

The superscript $(k-1)$ indicates that W has been determined from the residuals from the previous step. For the first iteration, the weight matrix is set to the identity matrix $W^{(0)} = I_m$.

Intuitively, Σ_{zz}' can be seen as the corrected covariance matrix of the measurements. It is clear that the variances of the correspondences just display the quality of the detected line feature, but not whether the feature was obtained from the right image feature or not. This means that the estimated variances are only valid, if they really belong to the right field lines. The weight matrix displays how good a measurement fits the observation model, i.e. exposes the confidence that a measurement was obtained by a correct feature.

The diagonal elements of the weight matrix are inferred from the proposed weight function (see section 4.2.5.4). The parameter a is selected as described in section 4.2.5.4, whereas b is set to 2 in the second iteration[12] and increased by 2 in each iteration.

If the initial estimate is close to the true state and the outlier rate is small, the final estimate is close to a weighted Gauss-Newton estimate with prior knowledge for outlier-free data. This means that the influence of the proposed weight function on inliers in the last iteration step is very small (see figure 4.3), while outliers are almost completely eliminated.

The iterations are terminated after the 3-th iteration or if the length of the gradient of the cost function (see section 4.2) falls below 10^{-6}. From the last iteration k', we obtain the final estimate for the current time t as

$$\hat{\boldsymbol{x}}_t = \boldsymbol{x}_{k'} \tag{5.96}$$

$$\hat{P}_t = \Sigma_{k'} \tag{5.97}$$

Figure 5.42 shows the first two iterations for the example depicted in figure 5.41.

[12]The weight matrix for the first iteration is set to identity.

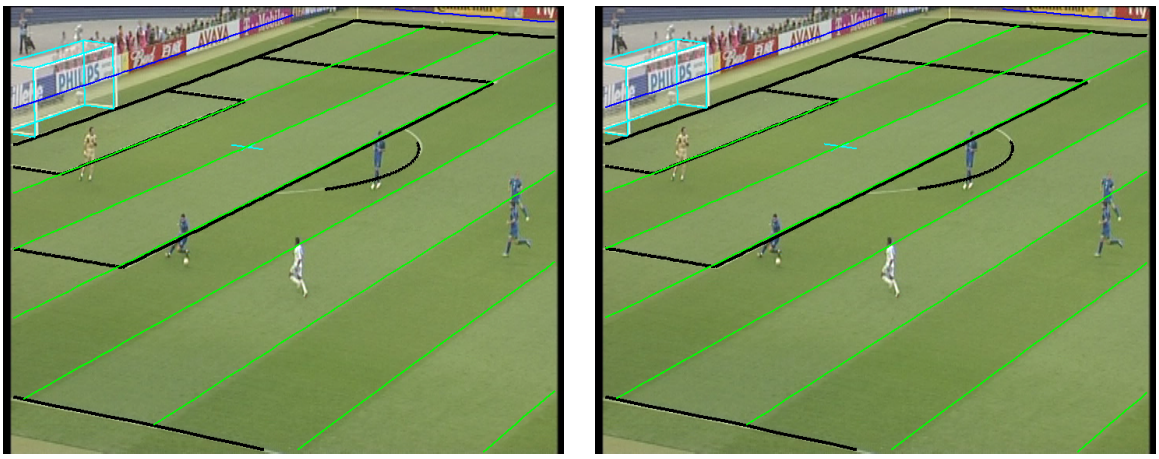


Figure 5.42 First and second iteration for example shown in figure 5.41.

5.6.4 Shot Boundary Detection

The shot boundary detection during the tracking task is carried out in the parameter prediction stage. As the sequences are taken from a rotating and zooming camera, we can exploit this knowledge to get a very robust shot boundary detection. Unlike the more generic methods described in section 5.4.1, that have to deal with arbitrary camera and object motion, we suppose that the transformation of two subsequent images can be described by a homography.

As described earlier, parameter prediction already determines the inter image homography between two subsequent images. As the initial guess for the homography is obtained using RANSAC, which also delivers the outlier rate for the best homography found, this information can be used for shot boundary detection. A shot boundary is detected if the outlier rate is above 80%. This holds for the case when no homography describing the change of two subsequent images accurately enough could be found.

The proposed method can reliably detect shot cuts as well as effects. Fortunately, during the active game, where camera tracking is performed, only shot cuts are used, and this method works very reliably. Besides the robustness, this method is also very efficient as it uses precomputed information from RANSAC. Table 5.16 shows the obtained precision and recall rates for this method. The ground truth data was manually obtained from several games.

	# seq.	false positives	false negatives	precision	recall
shot cuts	203	0	2	100%	99.01%

Table 5.16 Precision and recall rates for shot cut detection during the tracking task

If a shot boundary is detected, the tracking task returns the control back to the Scene Iterator module, which searches for the next suitable sequence.

5.6.5 Monitoring

Even though robust methods are used to overcome outliers in the measurements, and probabilistic methods are used to incorporate multiple uncertain measurements into one consistent and accurate estimate, camera tracking in natural and highly dynamic environments still stays a complex and error-prone task. Therefore, it is necessary to recognize if the tracker gets lost, i.e. the estimated parameters are far from the true state. A monitoring system runs during the update stage of the tracking task, and reinitializes the tracker if necessary. Three different methods are used, that complement each other.

The first method uses the initial fitness between the image and the model. For efficiency reasons and due to the high robustness of the tracking task (see section 5.6.6) no image lines are extracted, but instead the information gathered by the line correspondence search methods is used. As mentioned earlier, model lines with less than 3 found correspondences are marked as invisible. If the rate of "invisible" model lines exceeds 30%, we suppose that the parameters are wrong or out of the convergence range of the tracker.

The second method uses the final fitness between the model and the measurements, which is obtained by the weight matrix W (see section 5.6.3). As the diagonal elements of the weight matrix show the confidence that the corresponding measurement is an inlier, the "number" of inliers can be obtained as the sum of all diagonal elements $\mathrm{tr}\,(W)$ (trace of the weight matrix). If the inlier rate is below 80%, we suppose that the tracker got lost.

The third method uses the change of the parameters within two subsequent frames. Since TV broadcasts are meant for entertainment, camera rotation as well as the zooming speed is limited to obtain a smooth viewing experience. If one of these parameters changes too quickly, we suppose that the camera tracker gets lost. For the maximum allowed rotation speed we use $25\frac{^\circ}{sec}$ or 1° per frame. The threshold for change in the focal length is set to 1.1 per frame, which means that within two subsequent frames, the focal length can be increased by a maximum of 10% or decreased by a maximum of 9.1%.

If one of these methods detects the loss of tracking, execution is again passed to the *Initial Parameter Estimator*. If the *Initial Parameter Estimator* fails, the sequence is skipped and the control is returned to the *Scene Iterator* module. Otherwise, the tracker is reinitialized and processes the remaining parts of the video segment.

5.6.6 Experimental Results

The camera tracking was tested on several uncut video streams from TV broadcasting cameras. The average time (video time) before the tracker gets lost was approximately 19 minutes,

which is more than 29000 frames [13] .

We have evaluated the robustness and efficiency of the tracking task with the final game of the FIFA World Cup 2006. The video stream was grabbed by the main camera, which was placed in the center of the main stands at a height of $18m$.

The tracker processed both halftimes of the game without getting lost and, i.e. without any human intervention. The only two times the tracker gets lost was in the overtime. Both times, the camera zoomed in very fast and changed its orientation quickly during and after the zoom. Furthermore, no field markings were visible and a vast number of players in the view led to high outlier rates.

The camera tracking without player segmentation and player tracking was tested on an AMD Athlon 64 machine with 2GB RAM and running at 2200Mhz. The processing speed was about $15 - 20$ fps, which is close to real time. As the player regions, provided by the segmentation, are used to eliminate wrong feature correspondences, the average time until the tracker gets lost dropped to 14 minutes or 21000 frames [13] .

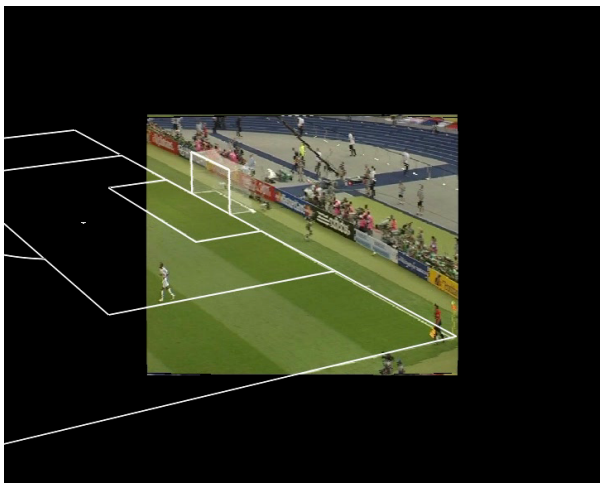
Furthermore, we have tested the tracker by supressing the update stage of the Kalman filter in frames where the predicted parameters were sufficiently accurate. The processing speed increased to 19 to 26 fps, while the accuracy was comparable to the normal method. The measurement update stage, which uses line correspondences, was only performed if the diagonal elements of the covariance matrix of the predicted parameters (their variances), exceeded some threshold values. By selecting a threshold value of 10^{-7} (radiant) for the rotation angles and $10^{-3}mm$ for the focal length, the measurement update stage was performed every 2nd to 5th frame.

The processing speed with player segmentation and player tracking was about $6 - 12$ fps, which is still very fast.

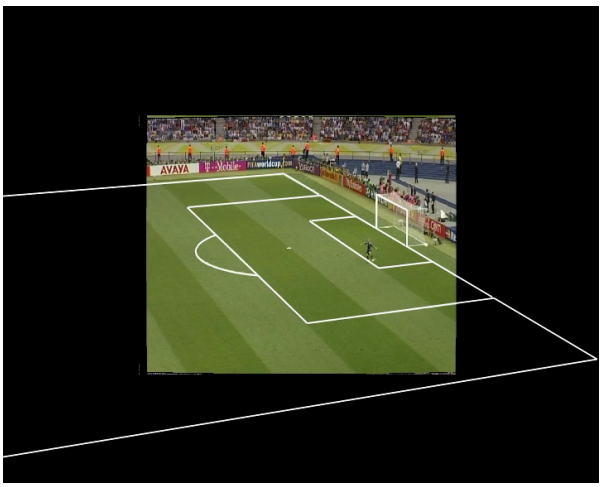
Additionally we have tested our methods successfully on broadcasted tennis videos, by only interchanging the field model. Since in tennis broadcasts the field is almost always completely visible, the number of used correspondences for each model line could be drastically reduced. We have used only three correspondences for each model line, which led to a higher processing speed. The robustness of the tracker could not be tested, as no uncut video streams were available.

Figures 5.43-5.45 show an example sequence for the robustness of the camera tracking system within the ASPOGAMO framework.

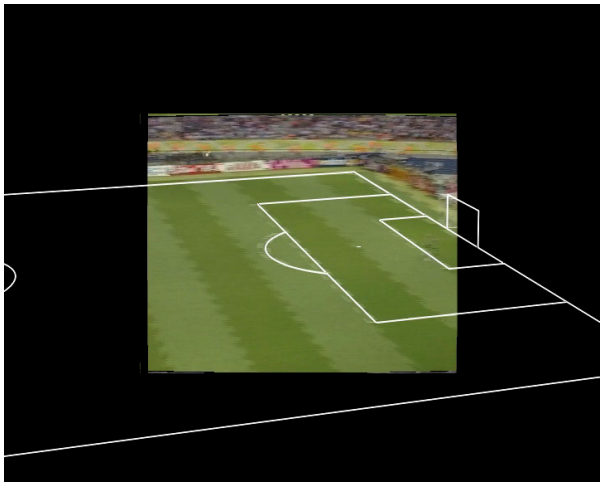
[13] PAL: 25fps



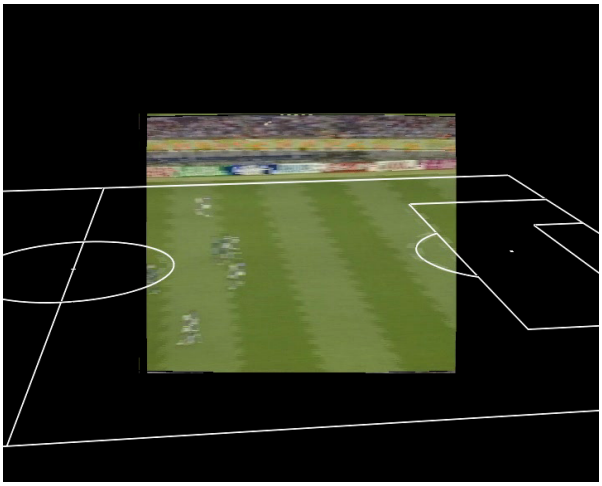
frame 21720



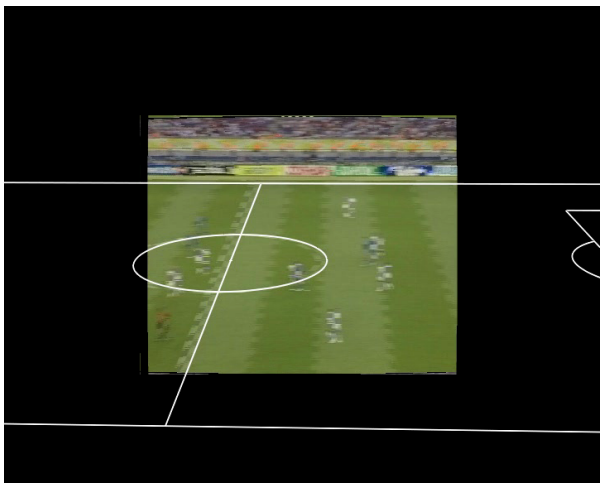
frame 22157



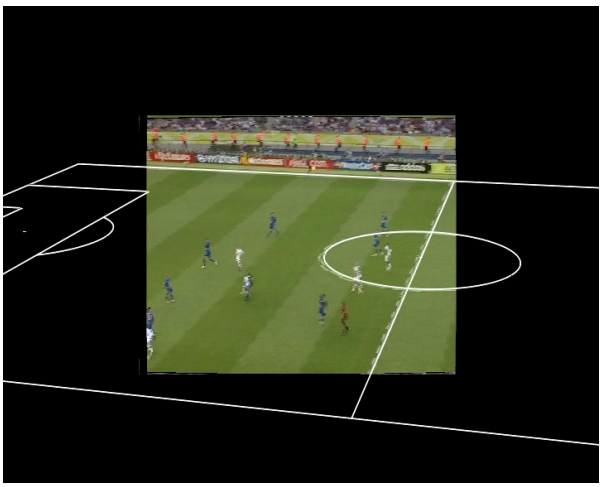
frame 22180



frame 22190

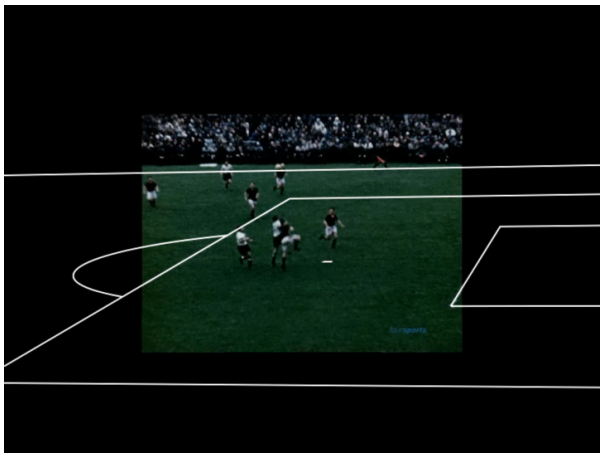


frame 22200

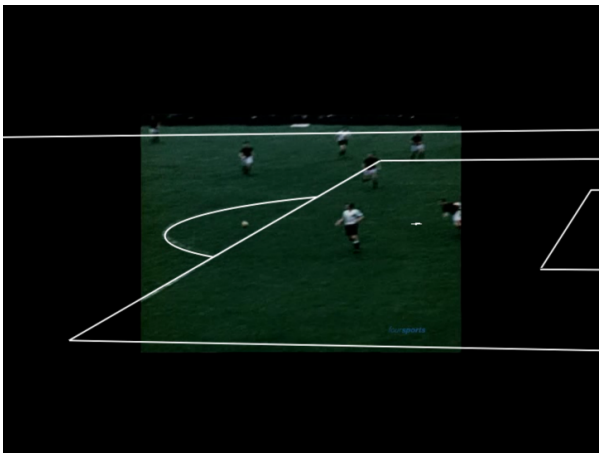


frame 22220

Figure 5.43 Successful tracking during fast camera motion.

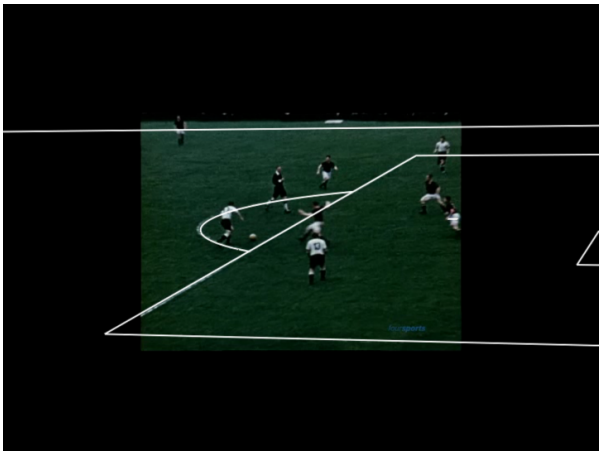


frame 0

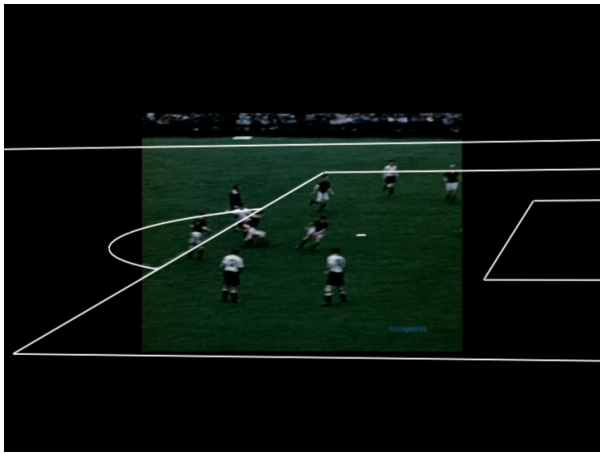


frame 30

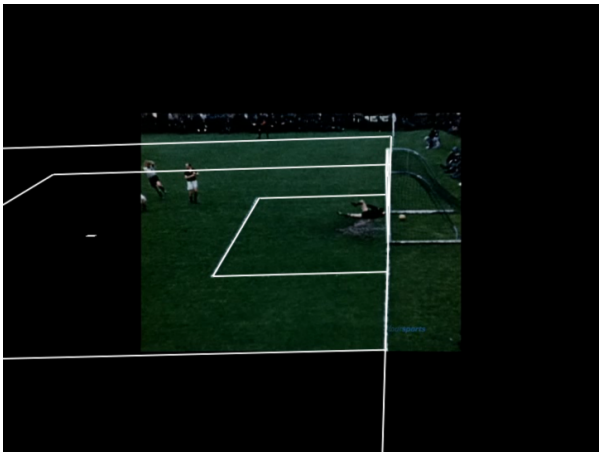
frame 60



frame 90

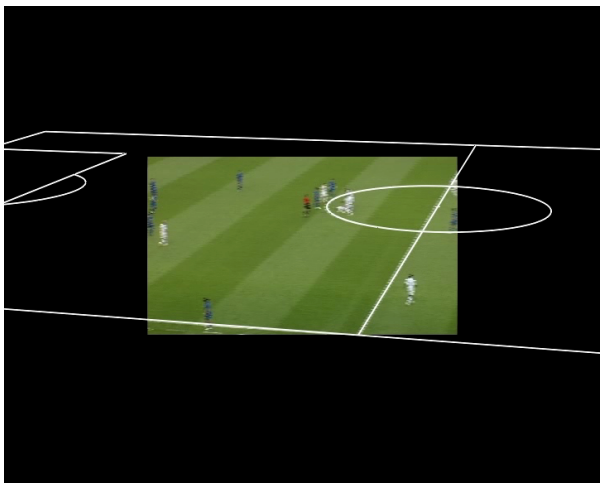


frame 120

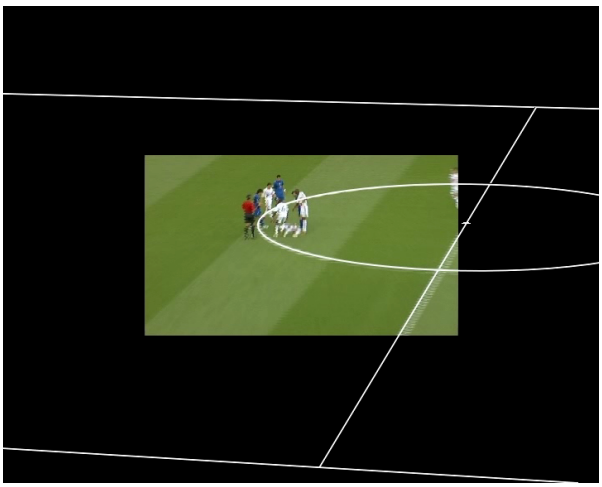


frame 150

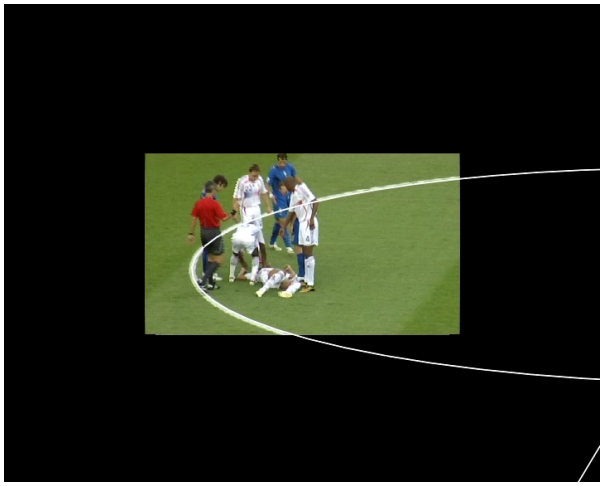
Figure 5.44 Successful tracking, even to suboptimal camera position (low height), pallid colors and flickering frames.



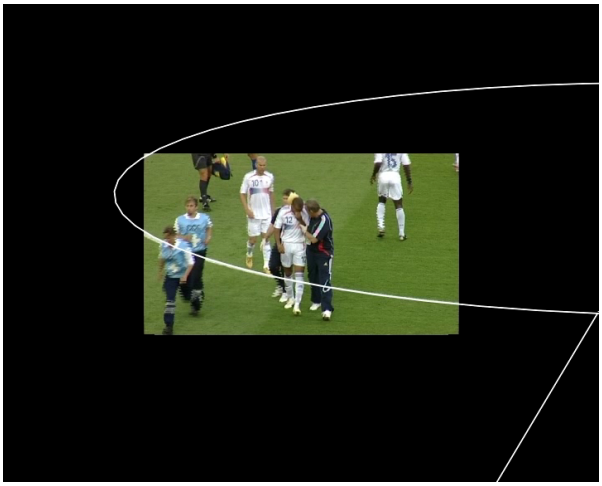
frame 1230



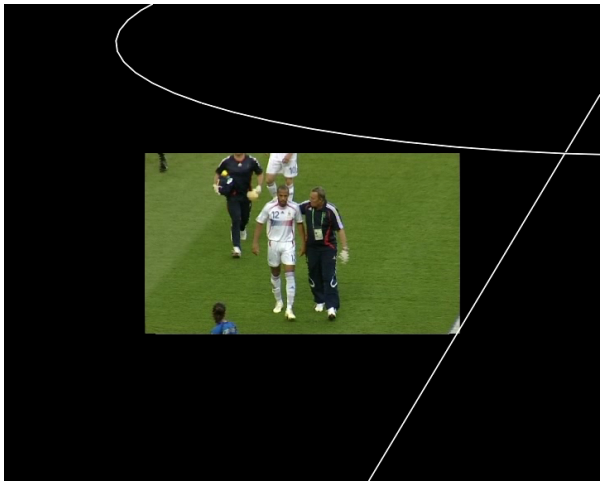
frame 1250



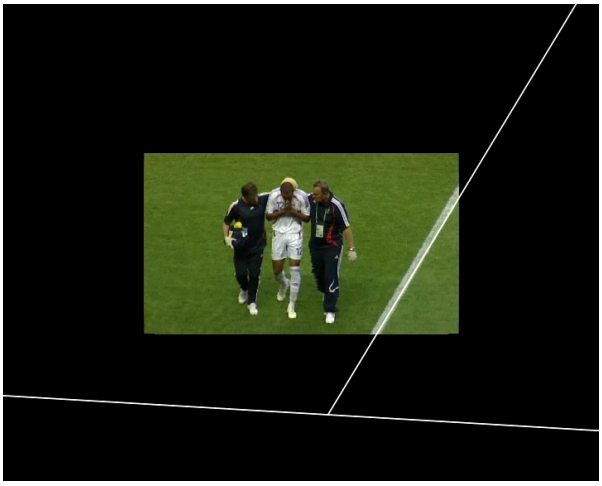
frame 1270



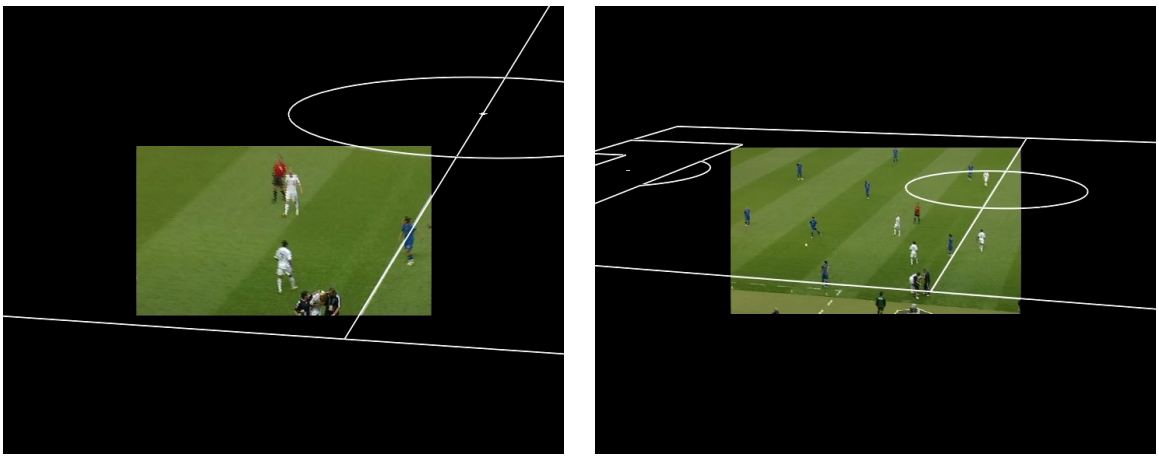
frame 2900



frame 4100



frame 4300



frame 4400 frame 4430

Figure 5.45 Successful tracking despite strong zooming and almost no visible field lines.

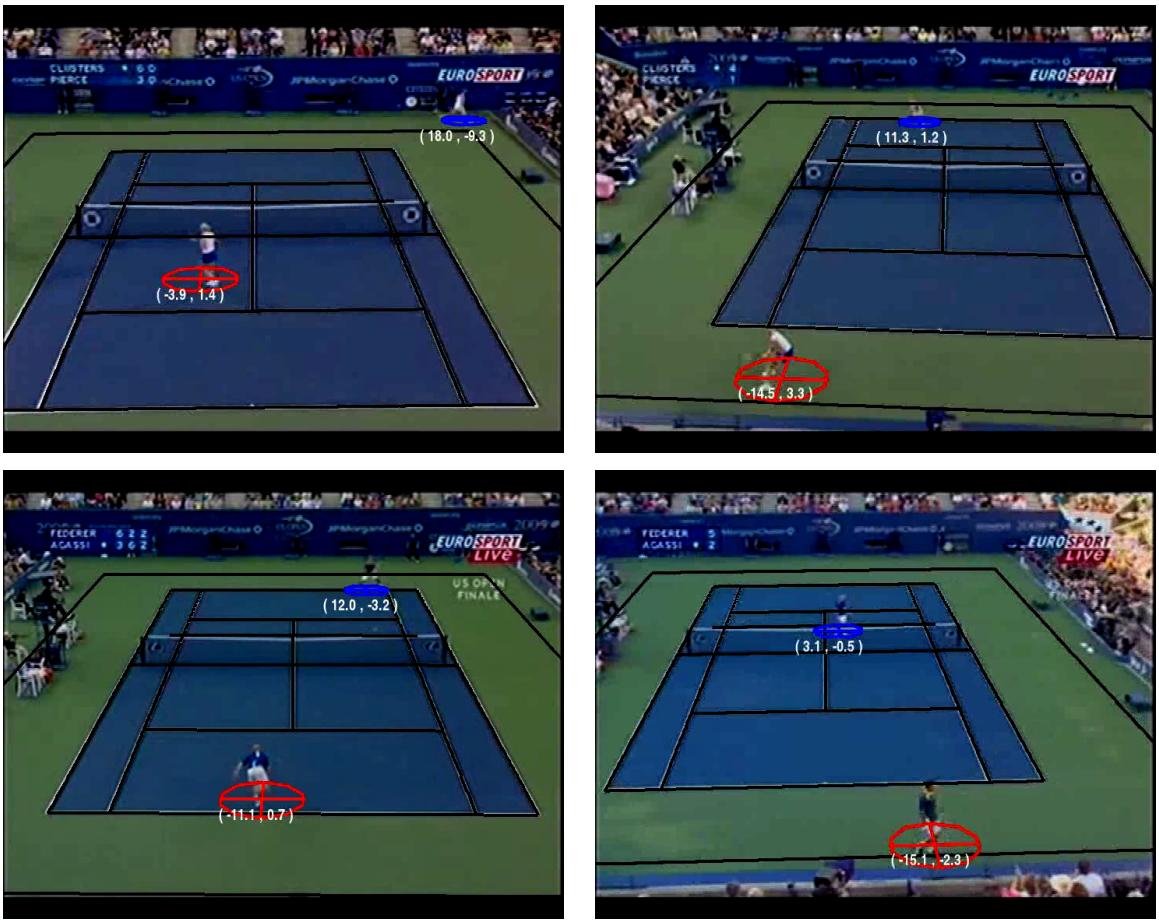


Figure 5.46 Successful tracking for broadcasted tennis videos

5.7 Related Work

To the best of our knowledge, so far no comparable system exists and therefore we will present related work integrated into constituent of our proposed system.

Farin et al. [FKEdW04, FHdW05] propose a model-based automatic camera calibration system for field sports, particularly for tennis. The camera calibration and tracking tasks are performed using image lines, which are extracted by a RANSAC-like line detector. The initial parameters are found by a combinatorial search of line correspondences. For efficiency reasons, they use two different methods to find line matches. The second method is slower but more robust than the first method, and is only carried out if the first one fails. The first method tests every combination of two parallel image lines with every combination of two parallel model lines, until a match is found. From each line correspondence, the homography is determined by using the correspondences defined by the four endpoints of the two line pairs. The model lines are projected into the image by using the homography, and the fitness is obtained by heuristics using the pixels classified as line candidates. As the first method is inappropriate for image lines which are only partly visible, the second matching method using four line correspondences is used. The number of combinations is reduced by selecting two horizontal and two vertical lines. For each set of line correspondences the homography is determined from the intersection points of the four lines.

The camera parameters are not determined explicitly, but are included in the homography. The tracking task predicts the camera parameters for the next time step by linear extrapolation using the last two time steps. From this prediction, the lines are extracted only in the vicinity of the expected positions, and the method described above is used to refine the parameters.

Besides the fact that the system can not process raw video streams from TV broadcasts, it has several other drawbacks. First, it does not exploit the domain knowledge completely. For example, the planar homography has 8 degrees of freedom and can describe the projection of the field for arbitrary camera configurations. However, as the camera setups in tennis (as well as in soccer) have fixed positions, the determination of the full homography is unnecessary and inefficient. Another drawback is that this method does not use the optical flow information for camera prediction and does not provide the uncertainty of the estimated parameters (homography).

Hayet et al. [HPV04a, HPV04b, HMC+05] propose a framework comparable to ASPOGAMO [BGB+07, GBvHH+07]. Like the method proposed by Farin et al. (see above), they do not estimate the camera parameters explicitly, but estimate the homography from the extracted image lines. However, they use the inter image homography for parameter prediction and pro-

vide the covariance matrix for the estimated parameters. The evaluation is just given for corner scenes and the method seems to work efficiently. Yet, it only works reliably on scenes with a sufficient number of visible field lines.

Yamada et al. [YSM02] propose a system for tracking the ball and the players in soccer and estimating the camera parameters. By using an initial guess, they project model lines into the image and search in its vicinity the nearest neighboring line pixel. The initial parameters for the first frame are obtained by a full search in the parameter space, while the predicted parameters are determined by a search only in the vicinity of the prediction. The predicted parameters for each time step are obtained from linear extrapolation.

However, the proposed method does not provide the uncertainty of the estimated parameters. Furthermore, a search in the parameter space is inefficient and leads to limited accuracy in the estimated parameters. Unfortunately, no experimental results were presented.

The camera calibration method proposed by Szenberg et al. [SCG01] uses a tree structure to find the right line matches. They extract image edges and filter out nonlinear structures by determining the eigenvalues within rectangular regions in the image. From the obtained straight edges, the field lines are recognized and identified by the so-called interpretation tree, where each leaf of the tree represents one possible combination of model lines. From the best match, obtained by the interpretation tree, the homography is determined and used to calculate point matches between the model and the image. In a succeeding step, these are used as the input for a standard calibration method to get the final camera parameters.

The method presented works reliably only if a sufficient number of field lines are visible in the image. Unfortunately, the presented results were obtained by small sequences (27 frames) displaying the penalty and goal area of the soccer field.

Watanabe et al. [WHK04] present a model-based camera calibration method using color information of the field and the field lines. Field lines are extracted by the field and line color information, which has to be provided by the user. The field size is supposed to be known, and the camera is supposed to be at the center of the main stands. The estimated parameters are therefore: pan and tilt angle, y and z coordinate of the camera, and focal length. The optimal parameters are found by a search in the parameter space around the initial guess, where the fitness of each parameter set is obtained by some heuristic function. The initial guess for each step is selected to be the estimate of the previous time step.

Bebie et al. [BB98] present a system called SoccerMan, which similarly to our system divides the camera parameters into a set of dynamic and constant parameters. They determine the constant parameters from the first frame of the video sequence, by manually identifying

image and model correspondences. The parameters of each frame are determined by tracking the lines between subsequent images.

Thomas et al. [JN06] partition the problem similar to our proposed framework into three stages. In the first stage, the position of the camera and its roll angle are determined by using multiple images. The second stage determines the initial parameters each time the tracking is initiated by a search of line correspondences in the Hough space. In the third stage, the actual camera tracking is performed. From a predicted state, which is simply the previous estimate, line correspondences are sought in the adjacency of the expected line positions, and the final parameters are estimated by the Levenberg-Marquardt method.

Intille et al. [Int94] present a system, that is capable to track football players from uncut video streams taken by rotating and zooming cameras. The player positions are obtained by using the homography between the image and the field, which is found by using at least four point correspondences. The point correspondences for the first frame are obtained manually, whereas the correspondences for the successing frames are given by intersection points of extracted field lines. However, as the other homography-based methods described above, this method lacks in the exploitation of the camera configurations to stabilize the estimated mapping, i.e. fixed position, no change in the roll angle and some fixed intrinsic parameters.

Ekin et al. [ET03, ETM03, Eki03] present a system, consisting of several low level components, that is capable to detect relevant events and summarize soccer games from TV broadcasts. They present methods for detecting the dominant color, shot boundaries, slow motion sequences, the referee and the penalty box.

The dominant color is obtained by the mean value of each color component in the HSI color space, which are determined around their corresponding histogram peaks from multiple images. Field color pixels are detected by thresholding their distance, which are determined by the cylindric metric, to the mean color.

The shot boundary detection uses two different features to reliably detect shot cuts as well as gradual transitions. The first feature is the absolute difference between two frames in their grass color ratios. The grass color ratio is obtained as the percentage of pixels classified as grass color. The second feature is the histogram difference between two frames, which is obtained by the histogram intersection. To reliably detect gradual transitions, the distance between the compared frames is set to 5 frames.

Video segments (shots) are classified into field view, close up view and out of field view by determining the grass color ratio within different regions in the frame. A Bayesian classifier using these features is designed to obtain the class of each frame in the video segment. The

class of the whole video segment is then chosen as the class of the majority of all frames.

Slow motion sequences are determined by the zero crossing measure (see section 5.4.3) and the penalty box detection is performed by simply searching three parallel lines within the field region, where two lines a supposed to be parallel if the angle between them is below a given threshold.

Chapter 6 Conclusion

In this thesis, we have investigated methods for an integrated vision system that is capable to estimate the parameters of dynamic cameras from live video streams of soccer broadcasts with high reliability. The results hereof are a set of powerful methods incorporated into a vision system that can perform the calibration and tracking tasks both accurately and reliably. The presented methods have been designed to overcome the noisy and erroneous measurements that are common in natural and highly dynamic environments.

The domain of team sport games has been chosen for the design and evaluation of such a system as it has a challenging yet well defined setting. It is also a promising area of application with high scientific and commercial potential.

We believe that we have made the following two main contributions to the field of camera calibration and tracking from TV broadcasts in team sports games:

1. We have presented an Iterative Extended Kalman Filter based approach to reliably track the camera parameters of rotating and zooming cameras over long periods of time. This is achieved using two techniques.

   First, camera parameters are predicted using optical flow information between subsequent images combined with robust estimation techniques. This is achieved without using any assumptions about the visibility of model features, by exploiting domain knowledge about the considered camera configurations. This allows the tracking task to bridge gaps, where no field lines are visible, by providing an accurate prediction for short time periods.

   Secondly, the MAP estimation during the update step which uses detected field line features in the image, is performed by a robust nonlinear least squares method. This is achieved by integrating the so-called M-Estimators into the commonly used weighted Gauss-Newton estimate. Furthermore, we use a proposed weight function that leads to a robust and accurate final estimate. It is achieved, by changing the shape of the weight function in addition to the clipping threshold in every iteration. In the final iterations,

outliers are almost completely eliminated, whereas inliers are almost equally weighted, so that the BLUE condition[1] of the estimate is almost unaffected by the weight function.

2. We have presented a vision framework that is able to handle raw video streams of soccer games from live TV broadcasts. It provides methods to determine constant camera parameters as well as the fieldsize and the field color from multiple views taken by the camera of interest. It is capable to identify significant sequences from the video stream by first segmenting the video at shot-boundaries and then classifying each sequence. Shots showing out of field views, close up views, or slow motion sequences are filtered as they contain no valueable information for the tracking task. For the remaining shots, the initial parameters of the camera are estimated using extracted image lines and knowledge about the camera position. The so-estimated parameters are used as an entry point for the camera tracker. We have showed that all of these methods work reliably and accurately, making the system applicable with only little human intervention.

The presented methods have been combined into a powerful integrated vision system running with near real time performance. It has been evaluated using TV-recordings gathered during the last FIFA World Cup in Germany. The camera tracking was extensively tested on several uncut video streams from different TV cameras with different perspectives, and the obtained results approved the accuracy and reliability of the proposed method. Camera parameters could be tracked for approximately 19 minutes of playing time or 29000 frames in average. Furthermore, both halves of the final game from FIFA World Cup 2006 in Germany were tracked without errors and without the need for reinitialization. Using only the initial prediction step from the camera tracker, the parameters could be predicted with sufficient accuracy for 2200 frames in average without the need for visible model features. The applicability of our methods for automated scene processing from live TV broadcasts of soccer games has also been shown. Segments showing slow-motion replays, out of field views or close up views could all be detected and distinguished with precision and recall rates clearly above 90%.

Despite the current focus on soccer games, we believe that our methods are also applicable in other and more general scenarios. The only requirements are cameras with a fixed position and a structured environment that can be described by a 3D model. Our methods would then be applicable by interchanging the model and incorporating suitable feature detectors. As an example, our system has been successfully tested on TV broadcasts of tennis matches. It turned out that camera tracking in tennis matches is easier and could be performed using much fewer

[1]The BLUE condition is given for linear observation models (see section 4.1).

correspondences due to the completely specified court sizes and the large amount of field lines visible in each frame. We are confident that other sports like volleyball or basketball would also be mastered by the presented methods.

Future research could aim at the automation of the last steps in the presented methods that require human intervention. As an example, the *Constant Parameter Estimation* module still requires some manually obtained point correspondences. These could possibly be extracted using methods similar to the ones for extracting and grouping image lines in the *Initial Parameter Estimation* module.

The *Initial Parameter Estimation* module itself could be improved to remove the dependencies on the amount of visible field lines. We propose to create a map of the environment using image mosaicking that would contain all available visual features and could be used for relocalization by deformable template matching if no line features are visible.

Bibliography

[AHC94] Farshid Arman, Arding Hsu, and Ming-Yee Chiu. Image processing on encoded video sequences. *Multimedia Systems*, 1(5):211–219, 1994.

[Ast65] Karl Astrom. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[Bar74] Yonathan Bard. *Nonlinear Parameter Estimation*. Academic Press, New York, 1974.

[BB98] Thomas Bebie and Hanspeter Bieri. Soccerman - reconstructing soccer games from video sequences. In *ICIP (1)*, pages 898–902, 1998.

[BBG+06] Michael Beetz, Jan Bandouch, Suat Gedikli, Nico von Hoyningen-Huene, Bernhard Kirchlechner, and Alexis Maldonado. Camera-based observation of football games for analyzing multi-agent activities. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.

[BBG+08] Michael Beetz, Jan Bandouch, Suat Gedikli, Nico von Hoyningen-Huene, Bernhard Kirchlechner, and Francisco Siles. Automated visual observation of soccer games from broadcasted videos. *Submitted to Computer Vision and Image Understanding*, 2008.

[BGB+07] Michael Beetz, Suat Gedikli, Jan Bandouch, Bernhard Kirchlechner, Nico von Hoyningen-Huene, and Alexander Perzylo. Visually tracking football games based on tv broadcasts. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[BGH+03] Michael Beetz, Suat Gedikli, Robert Hanek, Thorsten Schmitt, and Freek Stulp. AGILO RoboCuppers 2003: Computational Principles and Research Directions. In *RoboCup International Symposium 2003*, Padova, 2003.

[Bha43] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[Bou00] Jean-Yves Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker, 2000.

[BP99] Samuel Blackman and Robert Popoli. *Modern Tracking Systems*. Artech House, 1999.

[BR96] John S. Boreczky and Lawrence A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 170–179, 1996.

[BS89] Karl Brammer and Gerhard Siffling. *Kalman-Bucy-Filter*. R.Oldenbourg Verlag München Wien, dritte edition, 1989.

[BSF88] Yaakov Bar-Shalom and Thomas E. Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[Cai] Cairos. Cairos® *http://www.cairos.com/*.

[CH96] Ingemar J. Cox and Sunita L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(2):138–150, 1996.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley & Sons, second edition, 2000.

[EF04] Ralph Ewerth and Bernd Freisleben. Improving cut detection in mpeg videos by gop-oriented frame difference normalization. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 2*, pages 807–810, Washington, DC, USA, 2004. IEEE Computer Society.

[EHPM04] Jeff Erickson, Sariel Har-Peled, and David Mount. On the least median square problem. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 273–279, New York, NY, USA, 2004. ACM.

[Eki03] Ahmet Ekin. *Sports Video Processing for Description, Summarization, and Search*. PhD thesis, University of Rochester, 2003.

[ET03] Ahmet Ekin and A. Murat Tekalp. Robust dominant color region detection and color-based applications for sports video. In *Proceedings of the International Conference on Image Processing (ICIP)*, volume 1, pages 21–24, 2003.

[ETM03] Ahmet Ekin, A. Murat Tekalp, and Rajiv Mehrotra. Automatic soccer video analysis and summarization. *Image Processing, IEEE Transactions on*, 12(7):796–807, 2003.

[Fau93] Oliver Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[FB81] A. Martin Fischler and C. Robert Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[FHdW05] Dirk Farin, Jungong Han, and Peter H. N. de With. Fast camera calibration for the analysis of sport sequences. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2005.

[FHT96] Ludwig Fahrmeir, Alfred Hamerler, and Gerhard Tutz. *Multivariate Statistische Verfahren*. Walter de Gruyter, zweite edition, 1996.

[FIF07] FIFA. Laws of the game *http://www.fifa.com/*, 2007.

[FKEdW04] Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, and Peter H. N. de With. Robust camera calibration for sport videos using court models. In *SPIE Storage and Retrieval Methods and Applications for Multimedia*, 2004.

[FLB+04] Pascual Figueroa, Neucimar Leite, Ricardo M. L. Barros, Isaac Cohen, and Gerard Medioni. Tracking soccer players using the graph representation. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 787–790, Washington, DC, USA, 2004. IEEE Computer Society.

[FP03] David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[FW77] Mosteller F. and Tukey J. W. *Data Analysis and Regression - a second course in statistics*. Addison Wesley, 1977.

[GBvHH+07] Suat Gedikli, Jan Bandouch, Nico von Hoyningen-Huene, Bernhard Kirchlechner, and Michael Beetz. An adaptive vision system for tracking soccer players from variable camera settings. In *Proceedings of the 5th International Conference on Computer Vision Systems (ICVS)*, 2007.

[Gel74] Arthur Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[GZ05] Alexander Goldenshluger and Assaf Zeevi. The hough transform estimator. *Annals of Statistics*, 2005.

[Har97] Richard I. Hartley. Self-calibration of stationary cameras. *Int. J. Comput. Vision*, 22(1):5–23, 1997.

[HMC+05] J.B. Hayet, T. Mathes, J. Czyz, J. Piater, J. Verly, and B. Macq. A modular multi-camera framework for team sports tracking. In *IEEE International Conference on Advanced Video and Signal based Surveillance*, 2005.

[Hou62] Paul Hough. Method and means for recognizing complex patterns, 1962.

[HPV04a] J.B. Hayet, J. Piater, and J. Verly. Incremental rectification of sports fields in video streams with application to soccer. In *Accepted to IEEE Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS'04)*, 2004.

[HPV04b] J.B. Hayet, J. Piater, and J. Verly. Robust incremental rectification of sports video sequences. In *Accepted to British Machine Vision Conference (BMVC'04)*, 2004.

[Hub81] Peter J. Huber. *Robust Statistics*. Wiley-Interscience, 1981.

[HWJ94] A. Hampapur, T. Weymouth, and R. Jain. Digital video segmentation. In *MULTIMEDIA '94: Proceedings of the second ACM international conference on Multimedia*, pages 357–364, New York, NY, USA, 1994. ACM.

[HZ03] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[Int94] S.S. Intille. Tracking using a local closed-world assumption: Tracking in the football domain. In *Vismod*, 1994.

[IS03] Sachiko Iwase and Hideo Saito. Tracking soccer players based on homography among multiple views. In *SPIE*, pages 283–292, 2003.

[Jaz70] Andrew H. Jazwinski. *Stochastic Processes And Filtering Theory*. Academic Press, 1970.

[JEDGW81] Jr. John E. Dennis, David M. Gay, and Roy E. Walsh. An adaptive nonlinear least-squares algorithm. *ACM Trans. Math. Softw.*, 7(3):348–368, 1981.

[JHEJ98] Haitao Jiang, Abdelsalam Helal, Ahmed K. Elmagarmid, and Anupam Joshi. Scene change detection techniques for video database systems. *Multimedia Syst.*, 6(3):186–195, 1998.

[JN06] Sumit Jain and Ulrich Neumann. Real-time camera pose and focal length estimation. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 551–555, Washington, DC, USA, 2006. IEEE Computer Society.

[JS04] Florian Jarre and Josef Stoer. *Optimierung*. Springer Verlag, 2004.

[JU96] S. Julier and J. Uhlmann. A general method for approximating nonlinear transformations of probability distributions, 1996.

[JU97] S. Julier and J. Uhlmann. A new extension of the Kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, Orlando, FL*, 1997.

[KCM03] Jinman Kang, Isaac Cohen, and Gerard Medioni. Soccer player tracking across uncalibrated camera streams. In *PETS / ICCV*, 2003.

[KDD99] V. Kobla, D. DeMenthon, and D. Doermann. Detection of slow-motion replay sequences for identifying sports videos, 1999.

[KH90] Rakesh Kumar and Allen R. Hanson. Sensitivity of the pose refinement problem to accurate estimation of camera parameters. In *ICCV*, pages 365–369, 1990.

[KH94] Rakesh Kumar and Allen R. Hanson. Robust methods for estimating pose and a sensitivity analysis. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 60(3):313–342, 1994.

[KJ91] Rangachar Katsuri and Ramesh Jain. *Computer Vision: Principles*. IEEE Computer Society, 1991.

[Koc00] Ulrich Kockelkorn. *Lineare Statistische Methoden*. R.Oldenbourg Verlag München Wien, 2000.

[KS79] M. Kendall and A. Stuart. *The advanced theory of statistics. Vol.2: Inference and relationship*, volume 2. London: Griffin, 1979, 4th ed., 1979.

[LAF+93] Thomas D. C. Little, Gulrukh Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and Dinesh Venkatesh. A digital on-demand video service supporting content-based queries. In *ACM Multimedia*, pages 427–436, 1993.

[LBS01] T. Lefebvre, H. Bruyninckx, and J. De Schutter. Kalman filters for nonlinear systems: a comparison of performance, 2001.

[Leh86] E.L. Lehmann. *Testing Statistical Hypothesis*. Springer, 1986.

[Len87] Reimar Lenz. Linsenfehlerkorrigierte eichung von halbleiterkameras mit standarobjektiven für hochgenaue 3d-messungen in echtzeit. In *Mustererkennung 1987, 9. DAGM-Symposium*, pages 212–216, London, UK, 1987. Springer-Verlag.

[LK81] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[Low91] David G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13(5):441–450, 1991.

[LT88] R. K. Lenz and R. Y. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-d machine vision metrology. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):713–720, 1988.

[Lue91] Helmut Luetkepohl. *Introduction to Multiple Time Series Analysis*. Springer Verlag, 1991.

[Mar63] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

[May79] Peter S. Maybeck. *Stochastic Models Estimation, and Control*, volume 1. Academic Press, 1979.

[May82] Peter S. Maybeck. *Stochastic Models Estimation, and Control*, volume 2. Academic Press, 1982.

[MNO04] K. Madsen, H.B. Nielsen, and O.Tingleff. Methods for non-linear least squares problems, 2004.

[MT96] R. Mohr and B. Triggs. Projective geometry for image analysis, 1996.

[NPZ02] Chong-Wah Ngo, Ting-Chuen Pong, and Hong-Jiang Zhang. Motion-based video representation for scene change detection. *Int. J. Comput. Vision*, 50(2):127–142, 2002.

[NT92] Akio Nagasaka and Yuzuru Tanaka. Automatic video indexing and full-video search for object appearances. In *Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II*, pages 113–127. North-Holland, 1992.

[NW99] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 1999.

[PBS01] H. Pan, P. Van Beek, and M. Sezan. Detection of slow-motion replay segments in sports video for highlights generation, 2001.

[Pri] Simon J.D. Prince. Pattern recognition and machine vision: Kalman filter.

[Pro06] Ivan Stefanov Pronchev. Camera calibration in soccer from tv broadcasting. Master's thesis, Technische Universität München, 2006.

[PS] N. Patel and I. Sethi. Compressed video processing for cut detection.

[PS97] Nilesh V. Patel and Ishwar K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):583–592, April 1997.

[PTVF92] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[Ren98] Alvin C. Rencher. *Multivariate Statistical Inference and Applications*. John Wiley and Sons, 1998.

[Rou84] P.J. Rousseeuw. Least median of squares regression. *ASAJ*, 79:871–880, 1984.

[Rou87] P.J. Rousseeuw. *Robust Regression and Outlier Detection*. Wiley, 1987.

[Sca85] L.E. Scales. *Introduction to Non-Linear Optimization*. Palgrave Macmillan, 1985.

[SCG01] Flávio Szenberg, Paulo Cezar Pinto Carvalho, and Marcelo Gattass. Automatic camera calibration for image sequences of a football match. In *ICAPR*, pages 301–310, 2001.

[Sed92] Robert Sedgewick. *Algorithmen in C++*. Addison-Wesley, neunte edition, Januar 1992.

[Set03] Daniel Setterwall. Computerised video analysis of football - technical and commercial possibilities for football coaching. Master's thesis, CID, NADA, 2003.

[SGB04] Freek Stulp, Suat Gedikli, and Michael Beetz. Evaluating multi-agent robotic systems using ground truth. In *Proceedings of the Workshop on Methods and Technology for Empirical Evaluation of Multi-agent Systems and Multi-robot Teams (MTEE)*, 2004.

[Sha95] B. Shahraray. Scene change detection and content-based sampling of video sequences. In A. A. Rodriguez, R. J. Safranek, and E. J. Delp, editors, *Proc. SPIE Vol. 2419, p. 2-13, Digital Video Compression: Algorithms and Technologies 1995, Arturo A. Rodriguez; Robert J. Safranek; Edward J. Delp; Eds.*, volume 2419 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 2–13, April 1995.

[SKGB04] Freek Stulp, Alexandra Kirsch, Suat Gedikli, and Michael Beetz. AGILO RoboCuppers 2004. In *RoboCup International Symposium 2004*, Lisbon, July 2004.

[Sla99] Gregory G. Slabaugh. Computing euler angles from a rotation matrix, 1999.

[Sor70] Harold W. Sorenson. Least-squares estimation: from gauss to kalman. *IEEE Spectrum*, 7:63–68, July 1970.

[Sor80] Harold W. Sorenson. *Parameter Estimation. Principals and Problems*. Marcel Dekker, New York, 1980.

[SSJ93] D. Swanberg, C.F. Shu, and R.C. Jain. Knowledge-guided parsing in video databases. *SPIE*, 1908:13–24, February 1993.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[Str90] Peter Strohbach. *Linear Prediction Theory*. Springer Verlag, 1990.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[THF96] Tsuyoshi Taki, Jun-Ichi Hesagawa, and Teruo Fukumura. Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *ICIP*, volume 3, pages 815–818, September 1996.

[TM97] P. H. S. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *Int. J. Comput. Vision*, 24(3):271–300, 1997.

[TMHF00] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer-Verlag.

[Tor02] Wolfang Torge. *Geodäsie*. Gruyter, zweite edition, 2002.

[Tra] Trakus. Trakus® *http://www.trakus.com/*.

[Tsa87] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. pages 221–244, 1987.

[UMY91] Hirotada Ueda, Takafumi Miyatake, and Satoshi Yoshizawa. Impact: an interactive natural-motion-picture dedicated multimedia authoring system. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 343–350, New York, NY, USA, 1991. ACM.

[WCH92] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(10):965–980, 1992.

[Wer97] Joachim Werner. *Lineare Statistik*. Psychologie Verlags Union, 1997.

[WHK04] Tomoki Watanabe, Miki Haseyama, and Hideo Kitajima. A soccer field tracking method with wire frame model from tv images. In *ICIP*, pages 1633–1636, 2004.

[XOJ04] Ming Xu, James Orwell, and Graeme A. Jones. Tracking football players with multiple cameras. In *ICIP*, pages 2909–2912, 2004.

[YCK98] Yusseri Yusoff, William J. Christmas, and Josef Kittler. A study on automatic shot change detection. In *ECMAST '98: Proceedings of the Third European Conference on Multimedia Applications, Services and Techniques*, pages 177–189, London, UK, 1998. Springer-Verlag.

[YSM02] Akihito Yamada, Yoshiaki Shirai, and Jun Miura. Tracking players and a ball in video image sequence and estimating camera parameters for 3d interpretation of soccer games. In *ICPR (1)*, pages 303–306, 2002.

[Zha95] Zhengyou Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. Technical Report RR-2676, 1995.

[Zha00] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[ZKS01] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar. Automatic partitioning of full-motion video. In *Readings in multimedia computing and networking*, pages 321–339, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[ZLGS94] H. Zhang, C. Y. Low, Y. Gong, and S. W. Smoliar. Video parsing using compressed data. In S. A. Rajala and R. L. Stevenson, editors, *Proc. SPIE Vol. 2182, p. 142-149, Image and Video Processing II, Sarah A. Rajala; Robert L. Stevenson; Eds.*, volume 2182 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 142–149, March 1994.

[ZMM95] Ramin Zabih, Justin Miller, and Kevin Mai. A feature-based algorithm for detecting and classifying scene breaks. In *MULTIMEDIA '95: Proceedings of the third ACM international conference on Multimedia*, pages 189–200, New York, NY, USA, 1995. ACM.