Detection of Moving Objects by Spatio-Temporal Motion Analysis

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Clemens Rabe

Kiel

2011

Clemens Rabe
Detection of Moving Objects by Spatio-Temporal Motion Analysis

Detection of Moving Objects by Spatio-Temporal Motion Analysis

Dissertation

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
(Dr.-Ing.)

der Technischen Fakultät
der Christian-Albrechts-Universität zu Kiel

Clemens Rabe

Kiel

2011

1. Gutachter	Prof. Dr.-Ing. Reinhard Koch
2. Gutachter	Prof. Dr.-Ing. Joachim Denzler
Datum der mündlichen Prüfung	18.02.2011

Abstract

Driver assistance systems of the future, that will support the driver in complex driving situations, require a thorough understanding of the car's environment. This includes not only the comprehension of the infrastructure but also the precise detection and measurement of other moving traffic participants.

In this thesis a novel principle is presented and investigated in detail, that allows the reconstruction of the 3D motion field from the image sequence obtained by a stereo camera system. Given correspondences of stereo measurements over time, this principle estimates the 3D position and the 3D motion vector of selected points using Kalman Filters, resulting in a real-time estimation of the observed motion field. Since the state vector of the Kalman Filter consists of six elements, this principle is called 6D-Vision.

To estimate the absolute motion field, the ego-motion of the moving observer must be known precisely. Since cars are usually not equipped with high-end inertial sensors, a novel algorithm to estimate the ego-motion from the image sequence is presented. Based on a Kalman Filter, it is able to support even complex vehicle models, and takes advantage of all available data, namely the previously estimated motion field and eventually available inertial sensors.

As the 6D-Vision principle is not restricted to particular algorithms to obtain the image measurements, various optical flow and stereo algorithms are evaluated. In particular, a novel dense stereo algorithm is presented, that gives excellent precision results and runs at real-time. In addition, two novel scene flow algorithms are introduced, that measure the optical flow and stereo information in a combined approach, yielding more precise and robust results than a separate analysis of the two information sources.

The application of the 6D-Vision principle to real-world data is illustrated throughout the thesis. As practical applications usually require an object understanding rather than a 3D motion field, a simple yet efficient algorithm to detect and track moving objects is presented. This algorithm was successfully implemented in a demonstrator vehicle that performs an autonomous braking respifnextchar.. steering manoeuvre to avoid collisions with moving pedestrians.

Acknowledgment

The research presented in this thesis has been carried out at the research department Image based Environment Perception of the Daimler AG. The path towards this thesis spans several years of work within this department and many people have been involved and contributed to the presented ideas and understanding gained.

In particular, I wish to express my gratitude to my supervisors, Professor Dr.-Ing. Reinhard Koch and Dr.-Ing. Uwe Franke for their continued encouragement and invaluable suggestions during this work. I would also like to thank my second advisor Professor Dr.-Ing. Joachim Denzler for his time and interest, and the other members of my oral defense committee, Professor Dr. Manfred Schimmler, Professor Dr. Thomas Slawig and Professor Dr. Reinhard von Hanxleden.

Furthermore, I would like to thank my colleagues at the research team, namely Dr. Stefan Gehrig, Dr. Fridtjof Stein, Dr. Carsten Knöppel, Dr. Andreas Wedel, Dr. Alexander Barth, Heidi Loose, David Pfeiffer and Thomas Müller, for the productive discussions and new perspectives.

Finally, I wish to thank my parents, Sabine Kübler-Rabe and Dr. Udo Rabe, and all my friends for their never-ending support and continuous encouragement.

Mathematical Notation

Matrices

$\mathbf{M}$	A matrix consisting of $n \times m$ elements, organised in n rows and m columns, is written in non-italic boldface capitals.
$\mathbf{M}_{[i,j]}$	The element at the row i and column j of the matrix $\mathbf{M}$.
$\mathbf{I}_n$	The $n \times n$ *Identity Matrix,* whose n diagonal elements are set to 1 and the remaining elements set to zero.
$\mathbf{0}_{n \times m}$	The $n \times m$ *Zero Matrix* with all elements set to zero.
$\operatorname{diag}(a, b, c)$	The *Diagonal Matrix* with the given elements on the diagonal and the remaining elements set to zero.

Vectors

$\mathbf{v}$	Vectors are viewed upon as n-dimensional column vectors and are written non-italic lowercase boldface using commas to separate the elements: $\mathbf{v} = (a, b, c)^\top$
$\mathbf{0}_n$	The n-dimensional *Zero Vector* with all elements set to zero.

Decorations

$\widetilde{\mathbf{x}}$	The vector describes a point in the projective space or projective plane.
$\widehat{\mathbf{x}}$	The vector describes a predicted entity.
$\check{\mathbf{x}}$	The vector describes the undisturbed real values.
$\overline{\mathbf{x}}$	The mean of the random vector $\mathbf{x}$.

Coordinate Systems

$C(\mathbf{o}, \mathbf{x}, \mathbf{y}, \mathbf{z})$	A cartesian coordinate system with origin $\mathbf{o}$ and orthogonal unit vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ defining the x-, y- and z-axes.
$C_w(\mathbf{o}_w, \mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$	The fixed three-dimensional *World Coordinate System*. The subscript w is used for elements corresponding to this coordinate system.
$C_o(\mathbf{o}_o, \mathbf{x}_o, \mathbf{y}_o, \mathbf{z}_o)$	The three-dimensional *Observer Coordinate System*. It is fixed relative to a moving observer and is rotated and translated with respect to the world coordinate system. The subscript o is used for elements corresponding to this coordinate system.
$C_c(\mathbf{o}_c, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c)$	The three-dimensional *Camera Coordinate System*. Its origin $\mathbf{o}_c$ is the *camera centre* and it is rotated and translated with respect to the world coordinate system. The subscript c is used for elements corresponding to this coordinate system.
$C_r(\mathbf{o}_r, \mathbf{u}_r, \mathbf{v}_r)$	The two-dimensional *Retinal Coordinate System*. It is used to describe points projected onto the *retinal plane*. The coordinates of a point are given in meters. The subscript r is used for elements corresponding to this coordinate system.
$C_i(\mathbf{o}_i, \mathbf{u}_i, \mathbf{v}_i)$	The two-dimensional *Image Coordinate System*. It is used to describe projected points of the retinal plane in image coordinates. The coordinates are given in pixels. The subscript i is used for elements corresponding to this coordinate system.

Intrinsic Camera Parameters

f	The focal length in meter.
s_u, s_v	The width respifnextchar.. height of a pixel in meter.
f_u, f_v	The focal length in pixel defined as $f_u = \frac{f}{s_u}$ and $f_v = \frac{f}{s_v}$.
u_0, v_0	The principal point in pixel.
θ	The skew angle of the pixels.

Stereo Camera System Parameters

b	The base width in meters between the two camera centres of a standard stereo configuration.

Projection Matrices

$\mathbf{\Pi}$	The *Projection Matrix* to project a world point given in homogeneous coordinates onto the image plane.
$\acute{\mathbf{\Pi}}$	The *Extended Projection Matrix* to describe the projection of a world point given in homogeneous coordinates onto the coordinates $(u, v, d, 1)^\top$ of a standard stereo camera system. Here, u and v are the image coordinates in the left image and d is the disparity.

Probability Theory and Statistics Elements

$E[x]$, $E[\mathbf{x}]$	The expected value of the random variable x respifnextchar.. of the random vector $\mathbf{x}$.
$\mathrm{var}(x)$, σ_x^2	The variance of the real-valued random variable x.
$\mathrm{var}(\mathbf{x})$	The variance-covariance matrix of the random vector $\mathbf{x}$.
$\mathrm{cov}(x, y)$	The covariance of the real-valued random variables x and y.
$x \sim \mathcal{N}(\mu, \sigma^2)$	The real-valued random variable x is normally distributed with mean μ and variance σ^2.
$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$	The random vector $\mathbf{x}$ is normally distributed with mean vector $\boldsymbol{\mu}$ and the variance-covariance matrix $\mathbf{C}$.

Kalman Filter Elements

$\mathbf{x}(t)$, $\mathbf{x}_k$	The continuous-time *State Vector* at time t respifnextchar.. the discrete-time state vector at time step k.
$\widehat{\mathbf{x}}_k$	The *Predicted State Vector* for time step k.
$\mathbf{u}(t)$, $\mathbf{u}_k$	The continuous-time *Control Vector* at time t respifnextchar.. the discrete-time control vector at time step k.
$\boldsymbol{\omega}(t)$, $\boldsymbol{\omega}_k$	The continuous-time *Additive Noise Vector* of the system model at time t respifnextchar.. the discrete-time noise vector at time step k.

$\mathbf{z}_k$	The *Measurement Vector* at time step k.
$\boldsymbol{\nu}_k$	The *Additive Noise Vector* of the measurement model at time step k.
$\mathbf{s}_k$	The *Innovation Vector* at time step k.
$\mathbf{A}t$, $\mathbf{A}_k$	The *State Transition Matrix*.
$\mathbf{B}t$, $\mathbf{B}_k$	The *Control Matrix*.
$\mathbf{H}_k$	The *Measurement Matrix* at time step k.
$\mathbf{P}_k$	The *State Covariance Matrix* at time step k.
$\widehat{\mathbf{P}}_k$	*The Predicted State Covariance Matrix* at time step k.
$\mathbf{Q}t$, $\mathbf{Q}_k$	The covariance matrix of the continuous-time noise vector $\boldsymbol{\omega}(t)$ respifnextchar.. the discrete-time noise vector $\boldsymbol{\omega}_k$.
$\mathbf{T}_k$	The covariance matrix of the noise vector $\boldsymbol{\nu}_k$.
$\mathbf{S}_k$	The covariance matrix of the innovation vector $\mathbf{s}_k$.
$\mathbf{K}_k$	The *Kalman Gain*.
$\mathbf{V}_k$	The transformation matrix giving the influence of the additive measurement noise on the measurement for a non-linear measurement model.
$\mathbf{W}_k$	The transformation matrix giving the influence of the additive system noise on the predicted state for a non-linear system model.

Contents

1 Introduction **1**
- 1.1 Computer Vision in Driver Assistance 1
- 1.2 Detection of Moving Traffic Participants 3
- 1.3 Overview of the Proposed System 5
- 1.4 Organisation of the Thesis 8

2 Image Geometry and Correspondence Analysis **11**
- 2.1 The Pinhole Camera 11
- 2.2 Geometry of Two Views 16
  - 2.2.1 The Standard Stereo Configuration 16
  - 2.2.2 Rectification 18
  - 2.2.3 3D Scene Reconstruction 19
- 2.3 Correspondence Analysis 23
  - 2.3.1 The Correlation Stereo Algorithm 25
  - 2.3.2 The Kanade-Lucas-Tomasi Tracker 32

3 The 6D-Vision Principle **39**
- 3.1 Introduction 39
  - 3.1.1 Track-before-detect Motion Estimation 39
  - 3.1.2 The Need for Filtering 40
  - 3.1.3 Related Work 41
- 3.2 Kalman Filter based Motion Estimation 46
  - 3.2.1 The System Model 46
  - 3.2.2 The Measurement Model 49
  - 3.2.3 The Benefit of Filtering 49
  - 3.2.4 Mastering the Non-Linearity 51
  - 3.2.5 The Influence of Outliers 53
  - 3.2.6 Limitations 54
- 3.3 Multiple Filters for Improved Speed of Convergence 55
  - 3.3.1 The Initialisation Problem 55
  - 3.3.2 A Unified Decision Strategy 57
- 3.4 Conclusion 58

4 Ego-Motion Estimation **61**
4.1 Introduction . . . 62
4.2 Kalman-Filter based Ego-Motion Estimation . . . 63
4.2.1 The System Model . . . 63
4.2.2 The Measurement Model . . . 64
4.2.3 Selection of the Image Measurements . . . 67
4.2.4 Accuracy of the Estimated Ego-Motion . . . 68
4.3 Conclusion . . . 69

5 Variations of the 6D-Vision Principle **73**
5.1 Integration of Flow Fields . . . 73
5.1.1 Sparse Pixel-discrete Optical Flow Fields . . . 73
5.1.2 Dense Optical Flow Fields . . . 77
5.2 Dense Disparity Maps . . . 82
5.2.1 Semi-Global Matching . . . 82
5.2.2 TV-L^1 Stereo . . . 87
5.3 Exploiting the Epipolar Constraint . . . 89
5.3.1 Outlier Rejection . . . 90
5.3.2 Sparse Scene Flow . . . 91
5.3.3 Dense Scene Flow . . . 97
5.4 Conclusion . . . 101

6 Real-World Results **105**
6.1 Increased Robustness in Bad Weather Situations . . . 106
6.1.1 Dealing with Images of Low Contrast . . . 106
6.1.2 Coping with Rainy Weather Conditions . . . 108
6.1.3 Coping with Illumination Changes . . . 109
6.2 Object Detection and Tracking . . . 112

7 Conclusion **117**

A Discretization of a Continuous-Time System **121**

B Derivation of the Kalman Filter **127**

Bibliography **131**

List of Figures

1.1 Statistic of accidents in Germany from 1970 to 2007. 1
1.2 Stereo reconstruction of a typical traffic scene. 3
1.3 Optical flow field of a typical traffic scene. 4
1.4 Reconstructed motion field of the 6D-Vision algorithm. 5
1.5 Result of the image based ego-motion estimation. 6
1.6 Block diagram of the proposed system. 7

2.1 The ideal pinhole camera model. 12
2.2 World and camera coordinate systems. 14
2.3 Original and lens-corrected image. 14
2.4 Epipolar geometry and the standard stereo configuration. . . 16
2.5 Lens corrected stereo image pair. 18
2.6 Rectified stereo images. 18
2.7 Probability density function of the reconstructed depth and the resulting stereo bias. 20
2.8 Bias of the range with and without a second order correction. 21
2.9 Error intervals of the range PDF and the range bias introduced by the first order approximation of the error distribution. 21
2.10 Matching cost function and parabolic sub-pixel interpolation of the correlation stereo algorithm. 26
2.11 Ground truth stereo image pair used to evaluate the distribution of the measured disparities. 27
2.12 Distribution of the measured disparities using the parabola fit and the half shift method. 28
2.13 Distribution of the measured disparities using an iterative refinement based on the gradient-descent method. 29
2.14 Image 223 and 430 of the ground truth sequence used to evaluate the algorithms. 29
2.15 Ground truth disparity map and calculated disparity map of the correlation stereo algorithm. 30
2.16 Disparity error distribution of the correlation stereo algorithm. 32

2.17 Error distribution of the optical flow components estimated by the Kanade-Lucas-Tomasi tracker and evolution of the error over multiple frames. 37

3.1 Mean depth and mean depth velocity of a moving point. . . . 40
3.2 Mean depth and depth velocity of the proposed Extended Kalman Filter. 50
3.3 Measured and estimated state variance of the proposed Extended Kalman Filter for the depth and depth velocity. . . . 51
3.4 Error in the average state estimates for the depth and the depth velocity for the Extended Kalman Filter and the Iterated Extended Kalman Filter. 51
3.5 Variance of the average state estimates for the depth and the depth velocity for the Extended Kalman Filter and the Iterated Extended Kalman Filter. 52
3.6 Error in the average state estimates for the depth and the depth velocity for the presented Extended Kalman Filter and a modified version estimating the inverse depth. 53
3.7 Step response of the Extended Kalman Filter for a moving point. 54
3.8 Variance of the depth velocity of the differently parametrised Extended Kalman Filters of Figure 3.7. 55
3.9 Error in depth velocity for the same Extended Kalman Filter initialised with different velocities. 56
3.10 The values of the probability density function of the three differently initialised Extended Kalman Filters of Figure 3.9. 58

4.1 Observed 3D motion field and absolute motion field. 61
4.2 Estimation result of the 6D-Vision algorithm for two different ego-motion methods. 62
4.3 Problems of the classic $3\,\sigma$-test for the outlier detection. . . . 67
4.4 Result of the dynamic outlier detection based on the analysis of the normalised innovation squared. 68
4.5 Frame 93 and 104 of the synthetic sequence used to evaluate the performance of the proposed ego-motion estimation. . . . 69
4.6 Estimated angular and linear velocities of the proposed ego-motion algorithm. 70
4.7 Errors of the travelled distance of the ego-motion estimation. 71

5.1 Flow field of the PowerFlow algorithm. 74
5.2 Illustration of the spacial flow integration of a single feature using 5 patches. 75
5.3 Resulting feature tracks over the last 3 frames by the spacial integration of the PowerFlow field. 75

5.4 Error distribution of the optical flow components estimated by the PowerFlow tracker and evolution of the error over multiple frames. 76
5.5 Dense optical flow field obtained by the TV-L^1 algorithm. . . 77
5.6 Influence of the parameters λ and θ of the TV-L^1 optical flow algorithm on the average endpoint error and the RMS error. . 79
5.7 Error distribution of the TV-L^1 algorithm. 80
5.8 Evolution of the variance over multiple frames of the TV-L^1 algorithm using forward resp. backward calculation. 81
5.9 Disparity maps of the Semi-Global-Matching algorithms using ZSAD resp. Census matching costs. 82
5.10 Influence of the penalties of the Semi-Global-Matching algorithm on the average absolute error using ZSAD and Census matching costs. 84
5.11 Pixel-locking effect of the equiangular sub-pixel interpolation method of the Semi-Global-Matching algorithm using ZSAD and Census matching costs. 85
5.12 Error distribution of the Semi-Global-Matching algorithm using ZSAD and Census matching costs. 86
5.13 Disparity map of the TV-L^1 stereo algorithm. 87
5.14 Influence of the parameters λ and θ of the TV-L^1 stereo algorithm on the average absolute error and the RMS error. . . 88
5.15 Disparity and disparity error distribution of the TV-L^1 stereo algorithm. 89
5.16 Motion and stereo constraints of two stereo image pairs. . . . 89
5.17 Error distribution of the estimated scene flow components and evolution of the variances over time. 96
5.18 Utilised motion and stereo constraints by the proposed dense scene flow algorithm. 98
5.19 Optical flow field and disparity map obtained by the presented dense scene flow algorithm. 99
5.20 Error distribution of the optical flow and disparity components obtained by the proposed dense scene flow algorithm. . 100
5.21 Evolution of the variances over time. 101

6.1 Stereo camera system installed in a Mercedes S-Class. 105
6.2 Grey value distributions for a daylight and a tunnel scene. . . 106
6.3 Selected features of the KLT using the same threshold for the daylight and tunnel scenario. 107
6.4 Estimated motion field in the tunnel scenario. 108
6.5 Motion field estimation result in the presence of a wiper. . . . 108
6.6 Motion field estimation result after skipping the contaminated image containing the wiper. 109

6.7 Optical flow estimated by the Kanade-Lucas-Tomasi tracker during a change of the exposure time of the camera. 110
6.8 Optical flow estimated by the robust Kanade-Lucas-Tomasi tracker during a change of the exposure time of the camera, and the number of tracked features of the classic and the robust version over a complete sequence. 110
6.9 Error distribution of the optical flow components estimated by the robust Kanade-Lucas-Tomasi tracker and evolution of the error over multiple frames. 112
6.10 Illustration of the graph mapping and the result of the graph-cut segmentation. 113
6.11 Result of the risk analysis of the motion field. 114
6.12 Detected object of the object detection and tracking algorithm. 115
6.13 Illustration of the autonomous braking and evasion manoeuvre performed by the realised driver assistance system. 115

7.1 3D reconstruction and dense motion field estimated by the 6D-Vision algorithm for a pedestrian crossing the street. . . . 118

Chapter 1

Introduction

1.1 Computer Vision in Driver Assistance

Since the invention of the first gasoline-driven automobile by Carl Friedrich Benz in 1886, the number of cars world-wide increases constantly. In Ger-

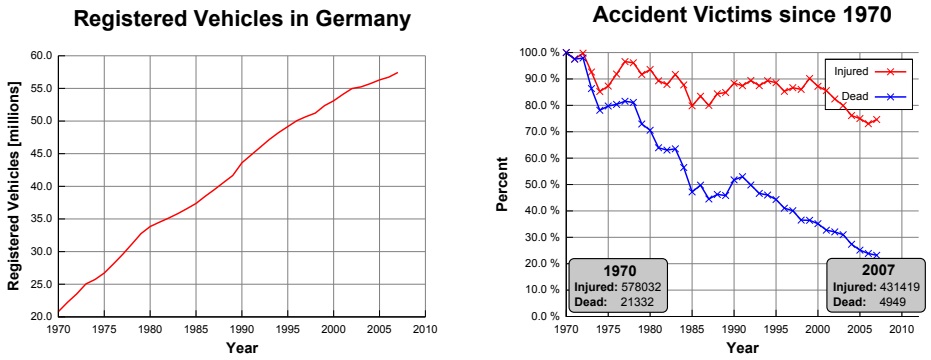

Figure 1.1: Registered vehicles in Germany (left) and evolution of accident victims (right) since 1970. (Data source [1]).

many, about 57.4 million vehicles were registered in 2007 and this number increases on average by over 1000 cars a day. Despite the continuously growing number of cars, as shown in the left hand diagram of Figure 1.1, the road safety improved over the last 30 years [1]. According to the current accident statistics, given in the right hand diagram of Figure 1.1, the number of people injured in an accident decreased since 1970 by more than 25 percent, and the number of deadly injured people decreased by about 76 percent. Besides improvements in the driver education and the increased regulation of the traffic, a main factor for the increased road safety are constructive safety measures. These are divided into active and passive systems. Passive safety measures, e.g., the airbag, increase the safety in the inevitable event of an accident. Active systems, like for example the Anti-lock Brake System,

take appropriate actions to prevent an accident or mitigate its impact.

In the last years the term *driver assistance system* was introduced to describe a group of active safety measures that assist the driver in critical situations. These systems are intended to increase the safety as well as the driving comfort by giving the driver additional information, like for example the Park Distance Control, or by taking corrective action, like the Adaptive Cruise Control, that adapts the speed of the car to ensure a safe distance to the preceding vehicle.

In general, a driver assistance system can be divided into three major blocks: The *environment perception* by analysing the data of various kinds of sensors to build a model of the environment, the interpretation of the current situation in a *situation analysis* step and a module executing the appropriate action, like warning the driver or an actuator performing a corrective action.

Depending on the main purpose of a driver assistance system, different kinds of sensors are used to perceive the environment. For example, the Park Distance Control uses multiple ultrasonic sensors embedded in the bumper to measure the distance to an obstacle, or the Adaptive Cruise Control uses a RADAR[1] sensor to measure the distance and speed of a preceding vehicle. Recent systems, like the Intelligent Headlight Control, the Lane Departure Warning or the Lane Keeping system, use video sensors to accomplish their goals.

In contrast to other sensors, like RADAR, ultra sonic or LIDAR[2], video sensors provide a high spatial resolution and allow the passive sensing of the environment at reasonable time intervals. Just like the human driver perceives its environment mainly through his eyes and analyses the visual information in the visual cortex, a driver assistance system uses *Computer Vision* algorithms to analyse the acquired images.

There are some challenging constraints for a computer vision algorithm in the context of driver assistance: First, the algorithm must cope with a large amount of data in real-time, that is at a frame rate of at least 25 frames per second (fps). Secondly, the algorithm must be robust in terms of contaminated image data, like for example dirt seen permanently in the image or a wiper covering a large part of the image. Last but not least, the algorithm has to provide some information about the uncertainty of the extracted information, that allows the following situation analysis to value and interpret the data correctly. The challenging task to provide rich and robust information in real-time is on the edge of current developments in computer vision.

[1] RAdio Detection And Ranging
[2] LIght Detection And Ranging

1.2 Detection of Moving Traffic Participants

Referring again to the current accident statistics, more than one third of all accidents with injuries occur at intersections, and most of them are caused by distraction, non-attention or misinterpretation of the situation [1]. The vast majority of these accidents is a collision with a moving object. To develop a suitable driver assistance system for such highly complex situations, that is able to warn the driver or even take corrective actions, a complete understanding of the traffic scene is required. Besides the perception of the infrastructure, like traffic signals, signs and the course of the road, the system must be able to detect other moving traffic participants and measure their movement precisely to predict potential collisions.

Video sensors are best suited for this complex task, and many algorithms exist for the perception of the infrastructure. For example, the detection of traffic signals and signs is a classical task of pattern recognition and is solved using means of classification [2, 3, 4], whereas the course of the road is estimated by analysing the road markings [5] or the free space in front of the vehicle [6]. However, the detection and measurement of moving objects under the mentioned real-time and robustness constraints remains a challenging task and is the focus of this thesis.

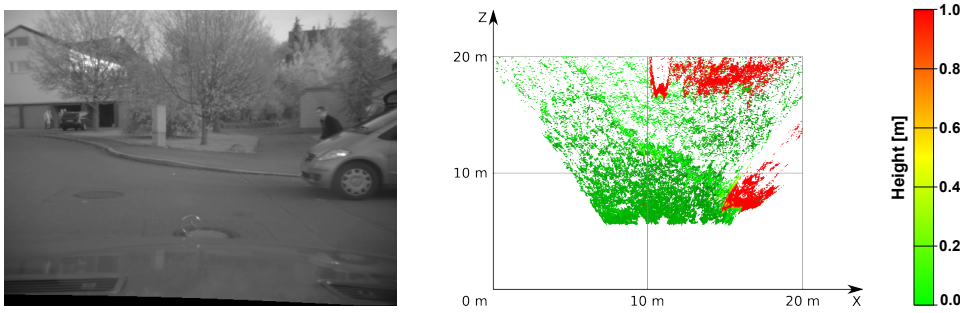

Figure 1.2: Left image captured by a stereo camera system (left) and the corresponding stereo reconstruction in a birds-eye view (right).

Combining two cameras in a stereo camera system, the three-dimensional structure of the scene is easily obtained by stereo vision algorithms. The result of such a reconstruction is illustrated in Figure 1.2: The left image shows a typical traffic scene captured by a stereo camera system: The observer moves towards an intersection at a constant speed of about 30 $^{km}/_{h}$, with a standing car at the right. Behind the car, a pedestrian moves towards the intersection. The right image shows the corresponding stereo reconstruction of that moment in a birds eye view. Here, the colour encodes the height of the reconstructed points above ground: Green encodes points on the ground, red points lie at a height of 1 m or higher. In this view,

obstacles can be detected as groups of protruding points, like for example the car on the left.

In fact, standard methods accumulate the stereo information in an evidence-grid like structure [7] and objects are then identified as regions of high spatial density in this map. To reveal the motion information, the detected objects are then tracked over time. The major disadvantage of such a *track-after-detect* strategy is that the performance of the detection depends highly on the correctness of the segmentation. For example, the moving pedestrian of Figure 1.2 cannot be distinguished from the standing car by analysing only the instantaneous stereo information. If the worst comes to the worst, the system merges the pedestrian and the car into a single static object and misses to warn the driver about the dangerous situation. Obviously, an analysis of the instantaneous stereo information is not sufficient for the detection of moving objects.

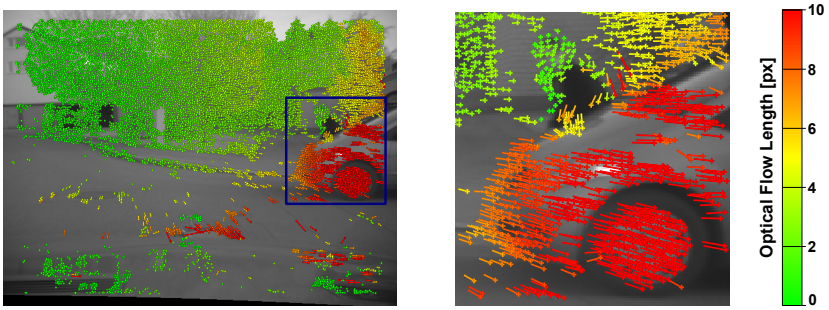

Figure 1.3: Optical flow field of the traffic scene shown in Figure 1.2.

Another group of Computer Vision algorithms for motion detection relies solely on the analysis of the apparent two-dimensional motion field seen in one camera. This motion field, also called the *optical flow*, depends mainly on the motion of the observer and the motions of the objects. Figure 1.3 shows the measured optical flow field for the same image of the traffic scene of Figure 1.2. The lines show the displacement vectors of the analysed image points, pointing from the position in the previous frame to the position in the current frame. The current position is also marked with a cross. The colour encodes the length of the displacement vector: Green encodes a small displacement vector, red encodes a displacement vector length of 10 px or more.

As the observer moves, all static world points induce a characteristic optical flow field that depends on the motion of the observer, also called the *ego-motion*, and the actual position of the world point. This image motion can be seen for example on the standing car or the buildings in the background. If the ego-motion is known, the three-dimensional structure of the static scene can be reconstructed solely from this optical flow field. This

is called *structure from motion* or *motion stereo* [8, 9, 10]. These algorithms can detect moving objects in some cases as a deviation from the expected flow field stemming from the static world assumption [9, 11]. However, a precise measurement of the world position and velocity is not possible using solely the optical flow field.

The optical flow field of moving objects depends not only on the motion of the observer, but also on the motion of the object. For example, the moving pedestrian in Figure 1.2 has small displacement vectors, as the observer and the object move at almost the same speed. Notice that in contrast to the stereo analysis of a single frame, the pedestrian and the standing car can be easily distinguished in this optical flow field. However, since the optical flow reveals no range information for the moving objects, their 3D-motion cannot be reconstructed solely from the optical flow field.

Obviously, a robust system for the detection and precise measurement of moving objects must combine the advantages of both fields, i.e., the stereo and the optical flow analysis. The stereo analysis reconstructs the world position of a point at a single time instance, whereas the optical flow gives the correspondence over time. Combining both information leads to a three-dimensional motion vector of the world point relative to the observer. In image space, this combination is called *scene flow* [12], and describes analogical to the optical flow a three-dimensional motion field. This flow information is calculated and analysed between two consecutive frames, yielding a differential information. However, to be more robust against the immanent measurement noise and to increase the sensitivity of the system, an integration of the differential information over time is required. This integration of the spatio-temporal information leads to the so called 6D-Vision principle that is presented in this thesis.

1.3 Overview of the Proposed System

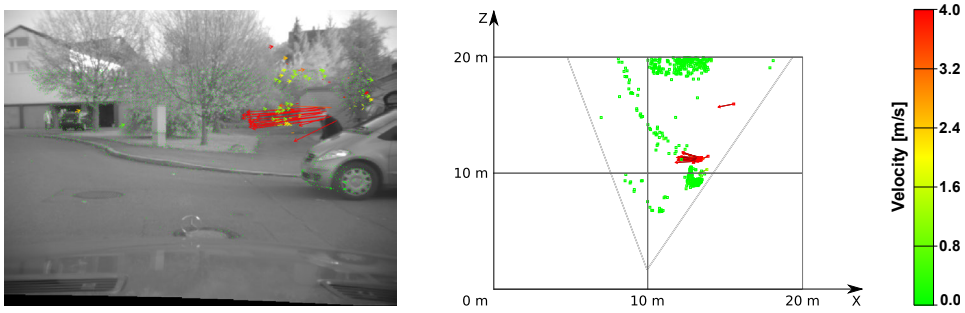

Figure 1.4: Reconstructed motion field of the 6D-Vision algorithm for the traffic scene shown in Figure 1.2.

In this thesis, the application of a stereo camera system for the real-time detection and tracking of moving objects from a moving observer is investigated. Its core algorithm, called *6D-Vision*, bases on the combination of the stereo information and the optical flow data to reconstruct the three-dimensional motion field along with the three-dimensional structure of the scene. In order to gain robustness, multiple measurements are integrated over time in a Kalman Filter system. The name 6D-Vision derives from the six-dimensional state vector estimated by the Kalman Filter. It consists of the three-dimensional position and its three-dimensional motion vector. The estimated three-dimensional motion field of the 6D-Vision algorithm is shown in Figure 1.4. The left image shows the estimated motion vectors pointing from the current world position to the estimated world position in 0.5 s, reprojected into the current image. The colour encodes the lateral velocity: Green corresponds to a small lateral velocity, red encodes 4 $^{m}/_{s}$ or faster. In the right image, the scene is shown in a birds-eye view. In contrast to the stereo reconstruction shown in Figure 1.2, the moving pedestrian can be easily distinguished from the standing car. In addition, the 6D-Vision information allows a prediction of the movement, and therefore provides a basis for the further situation analysis including risk assessment and collision avoidance strategies.

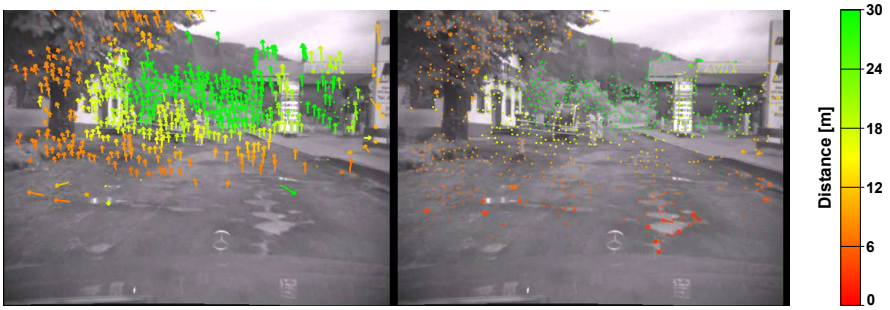

Figure 1.5: Reconstructed motion field using two different ego-motion estimation methods: In the left image, the ego-motion was derived from the inertial sensors (speed and yaw rate sensors). The right image shows the motion field of the same scene with the proposed image-based ego-motion estimation.

As the apparent motion field is composed of the motion of the observer, also called the *ego-motion*, and the motion of the observed objects, a system to detect moving objects must know the ego-motion exactly. Modern cars are equipped with inertial sensors, that can be easily used to determine the motion of the observer on the road. However, these sensors do not allow a full reconstruction of the three-dimensional ego-motion, since only a subset of the rotational velocities is available. For example, take a look at the left

image of Figure 1.5. Here, the car undergoes a heavy pitch movement which is not captured by a sensor, and can therefore not be compensated. As a result, the whole world seems to be moving.

As the motion of the observer induces an apparent motion field, it is possible to reconstruct the ego-motion solely by image analysis. In order to gain robustness, it is desirable to combine the measurements of the inertial sensors and the apparent motion field to get a precise ego-motion estimation. In this thesis, a new Kalman Filter based ego-motion estimation is presented, that takes advantage of the previous calculation steps and provides a fast and robust estimation of the observers motion. The result can be seen in the right image of Figure 1.5. In contrast to the left image, the stationary points are estimated correctly.

The resulting motion field encodes the motion of individual world points. To detect moving objects, this motion field is segmented, and identified objects are then tracked over time. By integrating the information of multiple points over time, the accuracy of the object motion estimation is increased. In addition, high-order terms, like for example its acceleration, can be estimated. This allows a more precise prediction of the object's movement than it is possible for individual world points.

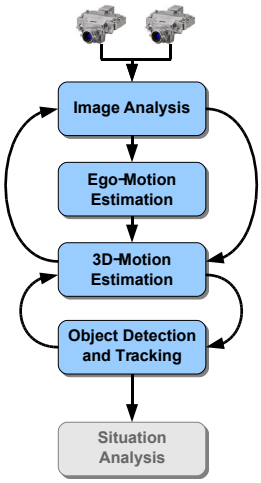

Figure 1.6: Block diagram of the proposed system.

Figure 1.6 shows the mentioned modules of the proposed system and the information flow. Starting with a pair of stereo images captured by a stereo camera system, the Image Analysis module retrieves the scene flow, that is the stereo information and the optical flow data. This information is used by the Ego-Motion Estimation module to determine the motion of the observer. The 3D-Motion Estimation Module integrates the optical flow and stereo information over time based on the 6D-Vision principle, and estimates

the 3D motion field, taking advantage of the ego-motion information. The calculated 3D motion field is then segmented in the Object Detection and Tracking module to detect new objects, and previously identified objects are verified and updated. The information about found and tracked objects is then given to the Situation Analysis module for further processing. However, a detailed description of this module is outside the scope of this thesis.

In order to interpret the provided information correctly, the situation analysis module requires also some uncertainty measures. Therefore, the shown information flow contains not only the information itself, but also its uncertainty.

In addition to the described information flow, there are optional feedback loops. Having the 3D motion field reconstructed for the previous time frame, it is possible to predict the image position in the current time frame by projecting the predicted world point position. This allows not only to reduce the search space in the correspondence analysis, but also leads to more robust input information, especially in the presence of repetitive patterns. Another feedback loop allows the combined estimation of the 3D motion field of points belonging to the same object.

1.4 Organisation of the Thesis

To clarify the mathematical notion and recall the basic terms and models of Computer Vision, a short introduction to the image geometry and the correspondence analysis is given in Chapter 2. This includes the introduction of two commonly used computer vision algorithms to determine the optical flow and the stereo information in real-time. In addition, the statistical properties of these algorithms are discussed.

The real-time reconstruction of the three-dimensional motion field is accomplished by the so called 6D-Vision principle. It fuses the stereo and optical flow information over time using a system of Kalman Filters and is investigated in detail in Chapter 3.

To distinguish the apparent motion field induced by a moving object from the motion field seen by a moving observer, the motion of the moving observer must be known. As the inertial sensors of common cars give only a subset of the required information to reconstruct the observers motion, an image based ego-motion algorithm is presented in Chapter 4. It takes advantage of the inertial sensors of the car and the information acquired by the previous analysis steps, i.e., the previous motion field, resulting in a robust estimation of the ego-motion.

The 6D-Vision principle is independent of the used stereo and optical flow calculation methods. Chapter 5 illustrates its application to variations of the input information, including the recently developed simultaneous real-time calculation of the optical flow and stereo information called *scene flow*.

The resulting motion field is then analysed with respect to moving objects. The detection and tracking of these objects is discussed along with the evaluation on real-world data in Chapter 6, followed by the conclusion and outlook in Chapter 7.

The main contributions of this thesis are the 6D-Vision principle (Chapter 3), the Kalman Filter based ego-motion (Chapter 4) and the novel scene flow algorithm (Section 5.3.3). Please note that parts of this work have been already published in [13, 14, 15, 16, 17, 18, 19, 20, 21].

Chapter 2

Image Geometry and Correspondence Analysis

The camera captures a two dimensional image of the three dimensional environment. A camera model describes the mathematical relationship between a three dimensional point and its projection on the image plane. Throughout this thesis, the ideal pinhole camera model is used, a simplified model for the *camera obscura*, i.e., the ancestor of the modern camera, that is introduced in Section 2.1.

By using two cameras, which capture the same scene from two slightly different points of view, the 3D structure of the scene can be immediately reconstructed. In Section 2.2, the geometrical properties of such a stereo camera system are discussed.

A more detailed introduction to the projective geometry, the pinhole camera model and the lens model is given in most computer vision books, such as, for example [22, 23, 24, 25, 8].

2.1 The Pinhole Camera

The ideal pinhole camera model describes the mathematical relationship between a three dimensional point $\mathbf{w}$ and its two dimensional projection $\mathbf{m}$ on the image plane. The aperture of the camera is assumed to be a point, and no lenses are used to focus the incoming light. Each ray reflected by an object of the scene passes through the pinhole and hits the image plane, resulting in the typical image of a central projection (see the left image of Figure 2.1).

Let us consider the geometric model of the ideal pinhole camera shown in the right image of Figure 2.1. The *camera coordinate system* $C_c\left(\mathbf{o}_c, \mathbf{x}_c, \mathbf{y}_c, \mathbf{z}_c\right)$ is a three dimensional right handed orthogonal coordinate system. Its origin $\mathbf{o}_c$ is the location of the *optical centre* or *camera aperture*. The $\mathbf{z}_c$ axis is pointing in the viewing direction of the camera, and is called *optical axis*,

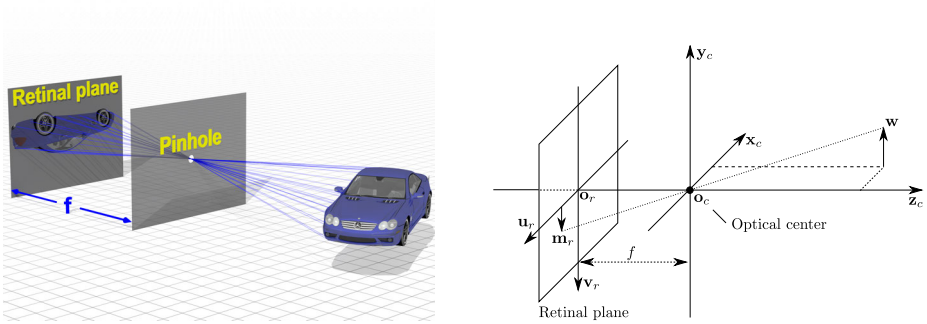

Figure 2.1: The ideal pinhole camera model. The left image shows the principle of the central projection obtained by a pinhole camera, the right image shows the geometrical relation of the projection of a single world point.

principal axis or *principal ray*. Light emitted or reflected by the world point $\mathbf{w}$ passes through the optical centre onto the retinal plane. The point of the intersection of the ray with the retinal plane is denoted $\mathbf{m}_r$.

The retinal plane is parallel to the $\mathbf{x}_c$ and $\mathbf{y}_c$ axes, and is located at the distance f from the camera coordinate system origin $\mathbf{o}_c$ in the negative direction of the $\mathbf{z}_c$ axis. The distance f is also called the *focal length*. To describe the projected point $\mathbf{m}_r$, we introduce the *retinal coordinate system* $C_r(\mathbf{o}_r, \mathbf{u}_r, \mathbf{v}_r)$. It is a two dimensional orthogonal coordinate system with its origin $\mathbf{o}_r$ at the point $(0, 0, -f)^\top$ expressed in camera coordinates. The origin is the intersection of the optical axis with the image plane, and is called the *principal point*. The axes $\mathbf{u}_r$ and $\mathbf{v}_r$ are parallel to the camera coordinate system axes $\mathbf{x}_c$ and $\mathbf{y}_c$, but point to the opposite directions.

The image sensor of a digital camera consists of a fixed number of picture elements, also called *pixels* or *pels*, organised in an array structure. Each pixel measures the *irradiance*, that is the amount of light energy accumulated over the sensor cells area in a fixed period of time, e.g., the shutter interval in a camera, or the integration time of a CCD array. The irradiance, often also referred to as the *image intensity* or *brightness*, is represented as a discrete integer whose range depends on the resolution of the A/D converter of the camera. Formally,

$$I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; (u, v) \mapsto I(u, v) \tag{2.1}$$

defines an image as a map I, defined on a compact region Ω of a two-dimensional surface, that takes values in the positive real numbers. In the case of digital images, both the region Ω and the value range $\mathbb{R}_+$ are discretized. For example, an image of a modern camera has a resolution of $\Omega = [0, 1023] \times [0, 511] \subset \mathbb{N}_0$ and the value range for an 12-bit image is approximated by an interval of integers of $[0, 4095] \subset \mathbb{N}_0$.

In projective geometry, the projection of a world point $\widetilde{\mathbf{w}} = (x, y, z, 1)^\top$ onto the image plane is given by the projection formula

$$\widetilde{\mathbf{m}}_i \simeq \mathbf{\Pi}\widetilde{\mathbf{w}}. \tag{2.2}$$

Here, $\widetilde{\mathbf{m_i}} = (u, v, 1)^\top$ denotes the projected point in homogeneous coordinates. Converted into the euclidean space, it gives the coordinates of the point in pixels. The 3×4 matrix $\mathbf{\Pi}$ describes a collineation of the projective space $\mathbb{P}_3$ onto the projective plane $\mathbb{P}_2$, and is called the *projection matrix* [22, 26]. It is composed of the *intrinsic parameters* encapsulated in the matrix $\mathbf{K}$, and the *extrinsic parameters* $\mathbf{R}$ and $\mathbf{t}$:

$$\mathbf{\Pi} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \tag{2.3}$$

Intrinsic calibration

The matrix $\mathbf{K}$ is given as

$$\mathbf{K} = \begin{bmatrix} \frac{f}{s_u} & \frac{f}{s_v}\tan\theta & u_0 \\ 0 & \frac{f}{s_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

with f as the focal length, s_u and s_v as the width respifnextchar.. height of a pixel, u_0 and v_0 as the coordinates of the principal point in pixels, and the skew angle θ. With $f_u = \frac{f}{s_u}$ and $f_v = \frac{f}{s_v}$, the focal length expressed in pixels, and the skew $s = f_v \tan\theta$, the matrix $\mathbf{K}$ becomes

$$\mathbf{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

It contains five parameters that are independent from the cameras position and orientation, and are therefore called *intrinsic parameters*. The matrix $\mathbf{K}$ is called the *camera calibration matrix*.

For a camera with a fixed optic, the intrinsic parameters remain constant over time. Therefore, these parameters can be obtained in an offline calibration step, usually together with the parameters of the lens model as described in the following. For dynamic optics, for example cameras with zoom or focus capabilities, the camera calibration matrix changes accordingly. A detailed analysis of these effects can be found in the work of Willson [27, 28, 29, 30].

Extrinsic calibration

The extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ describe the transformation of a point from the *world coordinate system* $C_w\,(\mathbf{o}_w, \mathbf{x}_w, \mathbf{y}_w, \mathbf{z}_w)$ into the camera coordinate

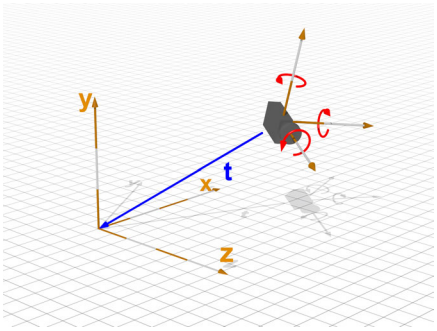

Figure 2.2: World and camera coordinate systems.

system. The point is first rotated using the rotation matrix **R**, and then translated by the translation vector **t**. Here, the translation vector **t** is given in the camera coordinate system pointing from the origin of the camera coordinate system to the origin of the world coordinate system, as illustrated in Figure 2.2.

Radial Distortion and Lens-Correction

Real lenses do not form perfect images, there is always some degree of distortion or aberration. There are different types of aberrations, for example spherical aberration, coma, chromatic aberration, astigmatism, or image distortion. Most of these types can be neglected under normal conditions, using high-quality optics. However, radial distortion can have a notable effect for shorter focal lengths. An example of radial distortion is shown in the left

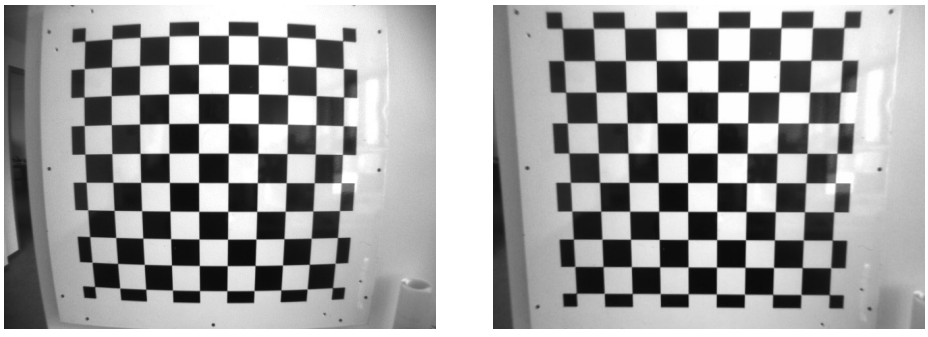

Figure 2.3: Image without (left) and with (right) lens-correction.

image of Figure 2.3. Here, the straight lines of the chequerboard do not result in straight lines in the captured image.

For most optics, the radial distortion can be separated into *radial-symmetric* and *radial-asymmetric and tangential* components [31, 32, 33]. The

relation between the distorted image coordinates $(u_d, v_d)^\top$ and the ideal coordinates $(u_c, v_c)^\top$ of the pin-hole camera model is given by

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u_c \\ v_c \end{pmatrix} + \Delta\mathbf{x}_r + \Delta\mathbf{x}_t \,. \tag{2.6}$$

$\Delta\mathbf{x}_r$ describes the radial-symmetric displacement caused by the fact, that world points at different angular distances from the optical axis undergo different magnifications. It is usually expressed as a polynomial

$$\Delta\mathbf{x}_r = \left(k_1 r^2 + k_2 r^4 + \ldots\right) \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix} \tag{2.7}$$

with the radius

$$r = \sqrt{(u - u_0)^2 + (v - v_0)^2} \,. \tag{2.8}$$

k_1 and k_2 are the coefficients of the radial-symmetric displacement.

The radial-asymmetric and tangential displacement is due to imperfect centring of the lens components and other manufacturing defects in a compound lens. It was first analysed by Brown [31] and is modelled as

$$\Delta\mathbf{x}_t = \begin{pmatrix} 2k_3\,(u - u_0)\,(v - v_0) + k_4\left(r^2 + 2\,(u - u_0)^2\right) \\ k_3\left(r^2 + 2\,(v - v_0)^2\right) + 2k_4\,(u - u_0)\,(v - v_0) \end{pmatrix} \tag{2.9}$$

with the coefficients k_3 and k_4.

The equations (2.6), (2.7) and (2.9) define a mapping of the captured, distorted image to the corresponding undistorted image (see the right image of Figure 2.3). This mapping is called *lens-correction* and allows the application of the ideal pin-hole camera model in the following image processing steps. For fixed optics, the coefficients k_i are determined in an offline calibration step, together with the intrinsic camera parameters (see [32, 33, 34, 35, 36]). With these parameters, the mapping is calculated once and stored in a look-up table, which is then applied to each distorted input image. The application of the look-up table usually involves a bi-linear interpolation for each pixel, that is efficiently implemented on a Central Processing Unit (CPU) using SIMD[1] instructions. More recently, Graphics Processing Units (GPUs) are used for this pre-processing task. Due to their massive parallel architecture and high memory bandwidth, they easily achieve a higher performance than a CPU based implementation. However, due to the overhead of the memory transfer, GPU based rectification modules are only faster when the images are large[2] or the rectified images are not required to be downloaded from the graphics memory.

[1] Single Instruction, Multiple Data (SIMD)

[2] The breakdown is at a resolution of about 1024×330 pixel when comparing an Intel Quad Core 3.0 GHz CPU with a NVidia GTX 285 GPU.

2.2 Geometry of Two Views

In a stereo camera system, two cameras capture images of the scene at the same time from slightly different point of views. In this section, the geometrical relationship between the two captured images is introduced first, and its impact on the scene reconstruction is shown. In this thesis, we concentrate on the standard stereo configuration, that allows important simplifications for the scene reconstruction algorithms, as described in Section 2.2.1. Since practical stereo configurations differ from this special configuration, a transformation process known as rectification is shortly explained in Section 2.2.2, followed by a discussion of the 3D scene reconstruction and its statistical properties in Section 2.2.3.

2.2.1 The Standard Stereo Configuration

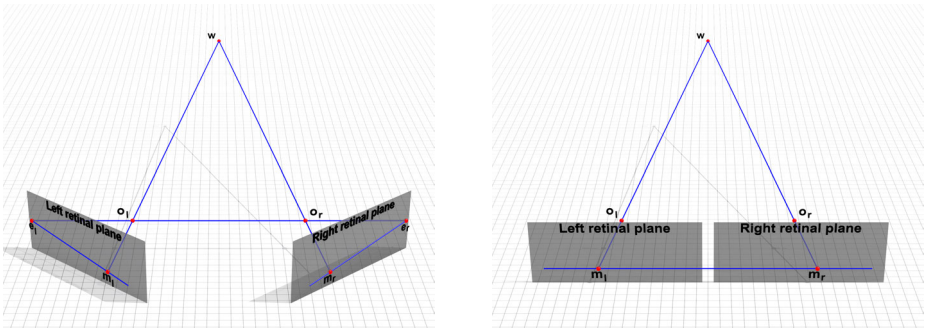

Figure 2.4: Epipolar geometry and the standard stereo configuration.

Having two pin-hole cameras, e.g., a left and a right camera, the projection of a world point $\mathbf{w}$ onto both image planes is given by the projection equation (2.2), resulting in the image points $\mathbf{m}_l$ in the left image and $\mathbf{m}_r$ in the right image. The intrinsic geometry of both perspective views, also called the *epipolar geometry*, is illustrated in the left side of Figure 2.4. For any world point $\mathbf{w}$, the plane defined by the projection centres, $\mathbf{o}_l$ and $\mathbf{o}_r$, and $\mathbf{w}$ is called the *epipolar plane*. The projections $\mathbf{m}_l$ and $\mathbf{m}_r$ also lie on this plane, since they lie on the rays connecting the world point and the corresponding camera centre. The projection of the epipolar plane onto the image planes gives the *epipolar lines* l_l and l_r. The line between the two camera centres is called the *baseline*, and it intersects the image planes in the points called the *epipoles*. So the left epipole $\mathbf{e}_l$ is the projection of the right camera centre $\mathbf{o}_r$ on the left image plane, and vice versa. All epipolar lines in one image intersect at the corresponding epipole.

For the scene reconstruction, the question is to find for one image point in one image the corresponding image point in the other image, and then

reconstruct the world point by a triangulation. For a given image point $\mathbf{m}_l$, the image point $\mathbf{m}_r$ corresponding to the same world point $\mathbf{w}$ has to lie on the epipolar line. This is also known as the *epipolar constraint*. Therefore, a stereo reconstruction algorithm has to search only along the epipolar line to find the corresponding image point $\mathbf{m}_r$ instead of performing a full two-dimensional search over the whole image, which reduces the computational complexity significantly.

If the image planes of both cameras are the same and the image coordinate systems are parallel to each other, the epipoles lie at infinity and the epipolar lines are parallel to each other. In addition, if the intrinsic parameters of both cameras are identical and the baseline is parallel to the image coordinates x-axis, the epipolar lines correspond to the same image rows in both images. This special case is called the *standard stereo configuration* and is shown in the right image of Figure 2.4. In this configuration, the correspondence search must be performed only along the image rows, that does not require additional sub-sampling of the image along angular epipolar lines and therefore simplifies the search significantly.

Taking the geometry of the standard stereo configuration into account, the relation between the x-coordinates of two corresponding image points $\mathbf{m}_l$ and $\mathbf{m}_r$ and the depth z_c of the world point in the camera coordinate system is given by

$$\begin{aligned} d &= u_l - u_r \\ &= \frac{b\, f_u}{z_c} \end{aligned} \tag{2.10}$$

Here, b is the distance between the two camera centres, and the *disparity* d denotes the difference between the x-coordinates of the left and the right image point. Combining this relation with the projection matrix $\mathbf{\Pi}$ of equation (2.3), the relation between the world point $\widetilde{\mathbf{w}}$ and the image coordinates $\widetilde{\mathbf{m}}_s = (u, v, d, 1)^\top$ is given by

$$\widetilde{\mathbf{m}}_s \backsimeq \acute{\mathbf{\Pi}} \widetilde{\mathbf{w}} \tag{2.11}$$

with $\acute{\mathbf{\Pi}}$ as the *extended projection matrix*

$$\acute{\mathbf{\Pi}} = \begin{bmatrix} f_u & 0 & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 0 & b\, f_u \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}. \tag{2.12}$$

Here, the skew parameter s of the corresponding matrix $\mathbf{K}$ defined by equation (2.5) was assumed to be 0. This is reasonable, as the standard stereo configuration is usually achieved by a pre-processing step called rectification, that transforms the original image pair to a common skew-free set of intrinsic parameters.

2.2.2 Rectification

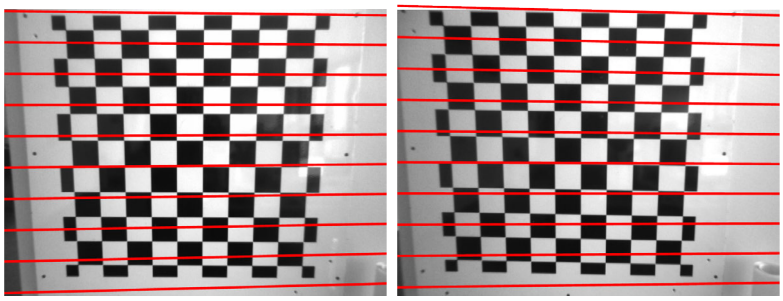

Figure 2.5: Lens corrected stereo image pair with the epipolar lines shown in red.

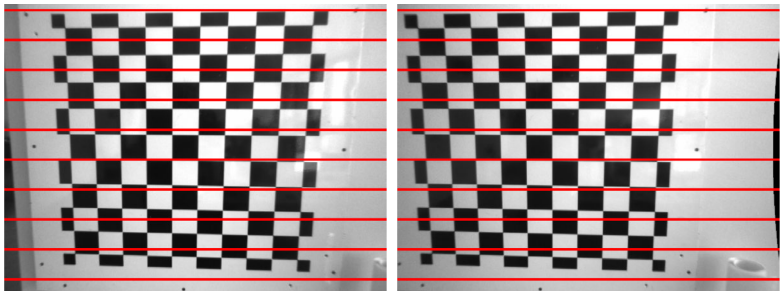

Figure 2.6: Rectified images of Figure 2.5. The epipolar lines are shown in red and correspond to the same epipolar lines as in Figure 2.5.

As shown in the previous section, the standard stereo configuration allows the simplification of the stereo reconstruction and reduces the computational complexity significantly. But in practice, an actual stereo camera system does not fulfil the requirements of the standard stereo configuration. However, it is possible to warp the original images to images corresponding to the standard stereo configuration. This process is called *rectification*.

In order to perform the rectification, the intrinsic and extrinsic parameters of the cameras must be known. These are usually obtained in an offline calibration process, using a calibration object with well known dimensions [34, 37]. To transform the original images into the corresponding images of the standard stereo configuration, the original images are projected onto two new image planes that fulfil the requirements of the standard stereo configuration [38, 39].

If the extrinsic camera parameters do not change over time, a look-up-table can be precomputed once for each camera, taking the lens-correction and the rectification steps into account. However, if the extrinsic parameters

change over time, for example in a system with cameras mounted on pan-tilt-units, it is more efficient to perform first the lens-correction using a look-up-table, followed by the re-projection of the images. This re-projection step is well-suited for GPUs, taking advantage of their massive parallel architecture.

Figure 2.5 shows a typical lens-corrected stereo image pair and the corresponding epipolar lines. After the rectification step, shown in Figure 2.6, the epipolar lines are parallel to each other and correspond to the same rows.

2.2.3 3D Scene Reconstruction

In general, to reconstruct the world point $\mathbf{w}$ from the image points $\mathbf{m}_l$ and $\mathbf{m}_r$ that fulfil the epipolar constraint, the intersection of the rays from the camera centres through the image points has to be calculated. If the camera matrices are known, that intersection point can be determined using basic linear algebra. In practice, camera parameters and image locations are known only approximately and the back-projected rays do not intersect at any point. Therefore, the world point $\mathbf{w}$ can only be estimated by minimising some error metric, usually the euclidean distance of the point to the rays. A more detailed analysis of this problem of triangulation is given in [40, 26, 41].

However, using the standard stereo configuration, the reconstruction of the world point from its measured image point $\widetilde{\mathbf{m}}_s = (u, v, d, 1)^\top$ is directly obtained by inverting equation (2.11):

$$\widetilde{\mathbf{w}} \backsimeq \acute{\mathbf{\Pi}}^{-1} \widetilde{\mathbf{m}}_s. \tag{2.13}$$

Although this equation is linear, the conversion of the homogeneous point $\widetilde{\mathbf{w}}$ into the euclidean space results in a non-linear relation.

Let us now focus on the uncertainty of the depth of the reconstructed world point. For a given image point $(u, v)^\top$ in one image, the stereo algorithm determines the disparity d by searching the corresponding point in the other image. As this is a measurement process, the relation between the true disparity $\check{d}$ and the measured disparity can be modelled using the additive noise ν as

$$d = \check{d} + \nu. \tag{2.14}$$

The probability density function (PDF) of the additive noise term ν is assumed to be symmetric, e.g., normal distributed, with a mean of 0 and a known variance σ_d^2. From equation (2.10) follows for the reconstructed depth of the point in the camera coordinate system:

$$z_c\,(d) = \frac{b\,f_u}{d} = \frac{b\,f_u}{\check{d} + \nu}. \tag{2.15}$$

Assuming the PDF of the measured disparity d is Gaussian, the probability

density function of the depth is then given by (see [42])

$$f(z) = \frac{b\, f_u}{\sqrt{2\pi}\sigma_d z^2} \exp\left(-\frac{\left(\frac{b\, f_u}{z} - \check{d}\right)^2}{2\sigma_d^2} \right). \tag{2.16}$$

This PDF is shown in the left image of Figure 2.7 for a given depth of

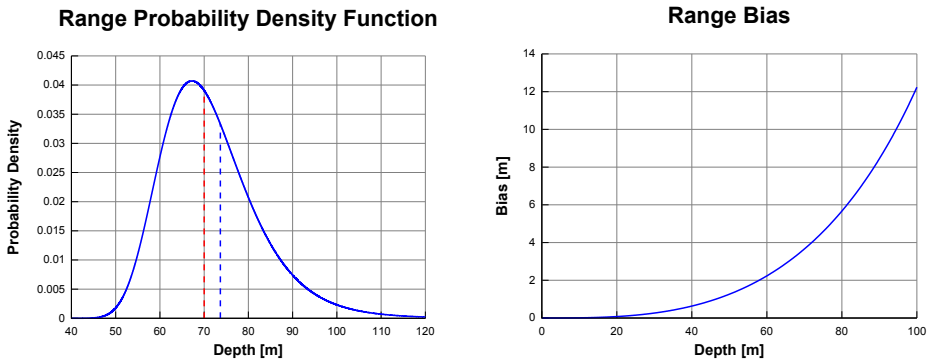

Figure 2.7: Probability density function of the reconstructed depth (left) and the resulting bias over range (right). The dashed red line marks the true depth of the point (70.0 m), the dashed blue line marks the mean depth of the PDF (73.7 m). (Parameters: $b = 0.30\,\text{m}$, $f_u = 800.0\,\text{px}$, $\sigma_d = 0.5\,\text{px}$)

70.0 m, that is marked with the dashed red line. Obviously, this function is asymmetric and its mean, marked by the dashed blue line, lies at 73.7 m. This difference between the real depth and the mean of a large number of depth samples originating from the same real depth results in a range bias, that increases with the depth (right image of Figure 2.7).

To reduce the effect of this bias, Sibley suggested in [42] to perform a second order expansion of equation (2.15) around the true disparity $\check{d}$:

$$z_c\left(\check{d} + \nu\right) \approx z_c\left(\check{d}\right) + \left.\frac{\partial z_c}{\partial d}\right|_{\check{d}} \nu + \frac{1}{2}\left.\frac{\partial^2 z_c}{\partial^2 d}\right|_{\check{d}} \nu^2. \tag{2.17}$$

With the expectations $E[\nu] = 0$ and $E\left[\nu^2\right] = \delta_d^2$, a better approximation of the depth $z_c\left(\check{d}\right)$ is then found as:

$$z_c\left(\check{d}\right) \approx z_c(d) - \frac{b\, f_u}{d^3}\delta_d^2. \tag{2.18}$$

The effect of this approximation is shown in Figure 2.8: The blue line shows the original range bias, and the red line shows the bias after using equation (2.18) on each sample. Obviously, the bias can be reduced significantly

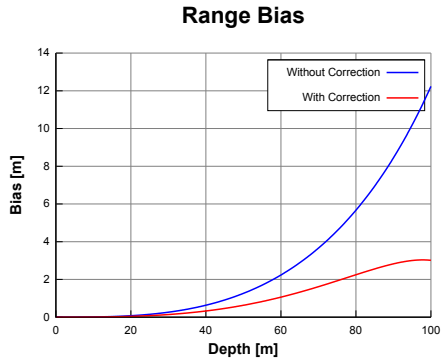

Figure 2.8: Bias of the range with (red) and without (blue) a second order correction.

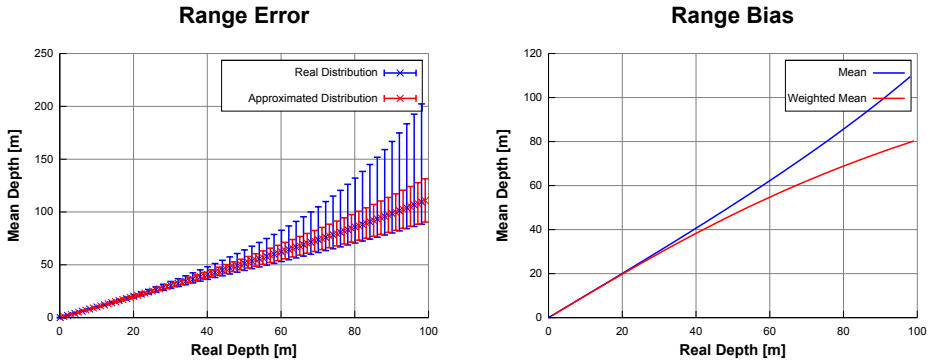

Figure 2.9: The left image shows the mean depth and the 15.9 % and 84.1 % percentile of the depth distribution (blue). The red error bars correspond to the approximated σ-range using classical error propagation. The right image shows the mean depth (blue) and the mean of the depth samples weighted with their inverse variance (red).

by this second order approximation when the disparity variance is known, but it can not be removed completely.

Estimation algorithms usually assume a symmetric PDF with a known scale parameter. As the relation between the disparity and the depth is non-linear, classical error propagation suggests to linearize equation (2.15) around the measured disparity and approximate the scale parameter by

$$\begin{aligned}\sigma_z^2 &= \left(\left. \frac{\partial z_c}{\partial d} \right|_d \right)^2 \delta_d^2 \\ &= \frac{b^2 f_u^2}{d^4} \delta_d^2. \end{aligned} \tag{2.19}$$

The approximated probability density function is then assumed to be symmetric around the mean $\bar{z} = \frac{b f_u}{d}$. In Figure 2.9 the 15.9 % and 84.1 % percentiles[3] of the actual depth distribution are shown in blue, whereas the σ-range of the classical error propagation is shown in red. Obviously, the approximated distribution slightly underestimates the lower bound, and clearly fails to match the upper bound.

The effect of using this approximation to estimate the mean depth is shown in the right image of Figure 2.9. Here, the mean depth weighting each depth sample the same is shown by the blue curve, and a weighted mean depth is shown by the red curve, with the weights calculated using the inverse variance of the depth derived from equation (2.19). The blue curve shows the already discussed bias, resulting in a mean depth that is greater than the real one. On the other hand, the weighted mean clearly underestimates the depth, because samples with a large disparity have a much higher weight than samples with a small disparity.

To eliminate this bias completely, the probability density function of the estimated parameters has to be accounted for in the estimation process. Although this is possible for the simple case of estimating a static point from a standing observer, this becomes nearly impossible for more complex scenarios, like for example a moving point seen by a moving observer. Therefore, depending on the problem and computational resources, non-linear estimation methods reduce the effect of the approximation error by an iterative refinement of the linearization point, e.g., the Iterative Extended Kalman Filter, re-sampling the probability density function using well-chosen sample points, e.g., the Unscented Kalman Filter, or by using Monte Carlo techniques, e.g., the Particle Filter.

[3]The 15.9 % and 84.1 % percentiles correspond to the $\pm\sigma$ range of a Gaussian distribution.

2.3 Correspondence Analysis

One of the main problems of Computer Vision is to establish correspondences between the projections of a world point in multiple images, for example the left and the right image of a stereo camera system or two consecutive frames of an image sequence. Once a correspondence is found, more specific geometric properties of the captured scene, like for example its 3D structure, can be reconstructed.

To establish a correspondence between a given image point $\mathbf{x}_1$ in one image I_1, the question is how to determine the position $\mathbf{x}_2$ in another image I_2, that belongs to the projection of the same world point. As both image points correspond to the same world point, their image coordinates are related by the so called *warp function*

$$\mathbf{x}_2 = h\left(\mathbf{x}_1, \mathbf{p}\right), \tag{2.20}$$

with $\mathbf{p}$ as a vector containing the unknown parameters. For example, in a standard stereo configuration, the parameter vector $\mathbf{p}$ consists of the *disparity* describing the translation along the x-axis. The function $h\left(\mathbf{x}_1, \mathbf{p}\right)$ is also referred to as the *image motion function*, as it can be seen as a function describing the motion of an image point from one image to another.

Throughout this thesis, we assume to deal only with Lambertian surfaces, that is the irradiance of a point is only determined by the direction of the surfaces normal to the light, and independent of the viewing angle of the observer [24, 43]. For cameras with the same imaging characteristics, this means that the irradiance of the projections in both images must be the same. This is commonly known as the *constant brightness assumption*, and the relation between the intensities is given by:

$$I_1\left(\mathbf{x}\right) = I_2\left(h\left(\mathbf{x}, \mathbf{p}\right)\right). \tag{2.21}$$

A naive attempt to search for correspondence would be to conclude from the constant brightness assumption, that a correspondence is found by searching for pixels of exactly the same irradiance. Obviously, this attempt has two major drawbacks: First, there will be most likely many pixels in the second image with the same intensity as the pixel in the first image. Secondly, the irradiance is a measured entity and therefore subject to measurement noise. It is also clear that equation (2.21) gives only one constraint for the parameter vector. If it consists of more than one parameter, for example a two-dimensional translation vector, the resulting equation system is underdetermined and a closed form solution can not be found.

One way to overcome the ambiguity of the correspondence is by defining a *support region* or *window* $W\left(\mathbf{x}\right)$ around the image point of interest. Assuming that all pixels undergo the same transformation, we formulate the equation system:

$$I_1\left(\mathbf{x}\right) = I_2\left(h\left(\mathbf{x}, \mathbf{p}\right)\right) \;\forall\, \mathbf{x} \in W(\mathbf{x}) \tag{2.22}$$

The position in the second image is then found as the position of the window with approximately the same intensity values for all points in the support region. Again, due to noise an exact match is unlikely. Therefore, instead of searching for a perfect match, the parameter vector is found at the point that minimises some discrepancy measure or error function ϵ between the reference window and all possible windows in the second image:

$$\arg\min_{\mathbf{p}} \varepsilon\left(I_1, I_2, \mathbf{x}, \mathbf{p}\right) \tag{2.23}$$

According to [44] ,the error functions found in the literature can be classified into the following categories:

- Cross-correlation-based measures
- Classical statistics-based measures
- Derivative-based measures
- Ordinal measures
- Robust measures

A typical error function of the class of classical statistics-based measures is the *sum of squared differences* (SSD). It measures the discrepancy of each pixel in the given window by calculating the difference of their intensities, which makes it sensitive to brightness changes between the two images. To overcome the influence of brightness variations, the *zero mean sum of squared differences* (ZSSD) measures the intensity relative to the mean intensity of the image patch. An extensive discussion on the various error functions found in the literature is given in the evaluations [45, 44].

To solve Equation (2.23) for a given image point $\mathbf{x}$, various error minimisation algorithms can be used. A naive approach is to perform a full search over the complete parameter space. Although this reveals the true minimum of the error function, the computational cost increases with the number of dimensions and the range of the parameter space. Therefore, minimisation algorithms, e.g., the gradient descent [46, 47], the Gauss-Newton method or the Levenberg-Marquardt algorithm [48, 49], which interpolates between the Gauss-Newton and the gradient descent, are used instead to determine a solution to the minimisation problem in reasonable time. The minimisation is usually performed assuming a local linearity of the image function and using the image gradient to determine the point minimising the linearised image function. As the image function is only approximated by a linear function, the algorithm has to be applied multiple times to find the solution. In addition, these algorithms do not search the full parameter space and can therefore not guarantee to find the global minimum.

Using a support region to find correspondences works only if the analysed patch has a texture of rich information. Imagine for example a region

of constant brightness. As the region is homogeneous, the exact position in another image can not be determined due to the *blank wall* or *aperture problem* [25]. Therefore, such a correspondence analysis is restricted to only those regions for which the correspondence problem can be solved unambiguously. Those regions are called *features* and are identified as regions of rich texture by a *feature detector* or *interest operator*.

Another way to solve the correspondence problem is to look at it as an optimisation problem. Instead of finding correspondences for the image points independently, global approaches determine the correspondences for all pixels of an image. They usually describe an energy or cost function, consisting of a data and a smoothness term. The data term measures the discrepancy between the solution and the actual image data, the smoothness term measures the local homogeneity of the solution. The global solution is then found by minimising this cost function. Typical examples are variational optical flow calculation [50], Semi-Global-Matching [51], belief-propagation or variational stereo calculation. Although these algorithms are computationally expensive, they recently achieve, implemented on dedicated hardware or on high-end graphic cards, real-time performance.

In the following sections, two widely used algorithms for the correspondence analysis are described in detail: The correlation-based stereo algorithm and the Kanade Lucas Tomasi tracker. Both algorithms can be used to calculate the input information for the 6D-Vision principle, that will derive the 3D motion field of the observed scene. In fact, these algorithms were actually used when developing the 6D-Vision method. As the determination of the 3D motion field is an estimation process, the uncertainties of the correspondences are of particular interest, and are investigated in the next two sections more closely.

2.3.1 The Correlation Stereo Algorithm

Stereo computation is a well investigated topic in Computer Vision, and many stereo algorithms are found in the literature. For an overview of current state of the art stereo algorithms, the reader is referred to [52] and the Middlebury website [53]. The vast list of algorithms can be divided into local methods, working on small image regions and producing either sparse or dense disparity maps, and global algorithms, that derive a global optimal solution for every pixel. Global algorithms calculate a dense disparity map by minimising an energy function, that consists of a data and a smoothness term. Both terms must be evaluated over the whole image region, and an update of the solution is usually found assuming local linearity of the energy function, which requires an iterative application of the method until convergence. Only parts of a global algorithm can be calculated in parallel, making it difficult to implement it on dedicated hardware. Therefore, local methods producing sparse disparity maps are often used in real-time applications, as

the computation time can be easily tuned by the variation of the number of correspondences to calculate. In addition, they are easily implemented on dedicated hardware, performing the computational expensive calculation steps in parallel.

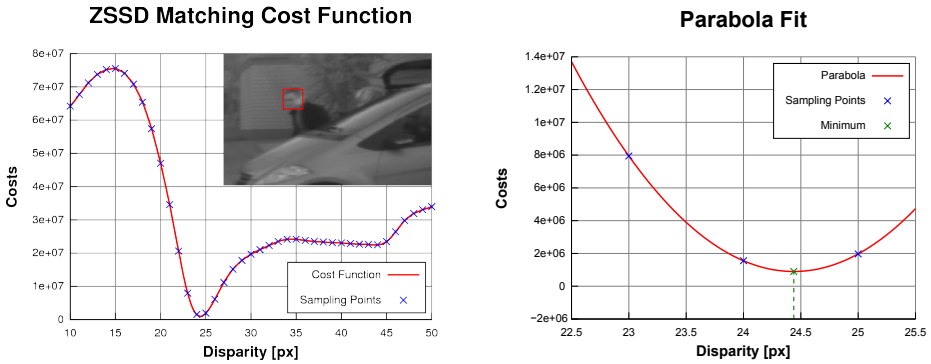

Figure 2.10: Matching cost function and parabolic sub-pixel interpolation of the correlation stereo algorithm.

A classical approach to determine the disparity map of a rectified stereo image pair is the *correlation stereo algorithm*. Using a pair of rectified images, the image motion function giving the image coordinates in the right image is defined as:

$$h\left(\mathbf{x}, d\right) = \mathbf{x} - \begin{pmatrix} d \\ 0 \end{pmatrix} \tag{2.24}$$

For a given image point $\mathbf{x}$ in the left image, the value of the error function $\epsilon\left(I_l, I_r, \mathbf{x}, d\right)$ is calculated over a predefined disparity range, as shown in the left image of Figure 2.10. Here, the result of the *zero mean sum of squared differences* (ZSSD) is given, that is defined as

$$\epsilon_{\text{ZSSD}}\left(I_l, I_r, \mathbf{x}, d\right) = \sum_{W} \left(\left(I_l\left(\mathbf{x}\right) - \bar{I}_l\right) - \left(I_r\left(h\left(\mathbf{x}, d\right)\right) - \bar{I}_r\right)\right)^2 \tag{2.25}$$

with $\bar{I}_l$ respifnextchar.. $\bar{I}_r$ as the mean intensity of the analysed window W. The ZSSD error metric is widely used, due to its improved robustness against brightness changes compared to the *sum of squared differences* or the *sum of absolute differences*. The disparity is then found at the minimum of the error function. Obviously, a distinctive minimum can only be found, if the analysed patch contains a distinctive image structure. Therefore, the correlation is only performed on regions of sufficiently high image structure, that are easily identified by thresholding the gradient of the image along the x-axis.

A common approach to speed up the correlation process is to introduce an image pyramid [54]. By evaluating the error function first on the image

pair with the lowest resolution, and propagating the minimum down to the next pyramid level, the error function has to be recalculated only around the already found minimum. For example, for a disparity range of $[0\,\text{px}; 127\,\text{px}]$ and an image pyramid containing two sub-sampled versions of the original images with a sub-sampling factor of 0.5, only 32 disparity steps have to be evaluated on the first image.

To gain robustness, additional constraints are usually accounted for in the correlation algorithm:

- The *smoothness constraint* is based on the fact that the disparity on surfaces changes only slightly, whereas it may change significantly on edges.
- The *uniqueness constraint* states that a world point has exactly one projection in each image. This can be easily exploited to remove wrongly estimated disparities in occluded areas, by re-evaluating the error function using the right image as the reference image. Occluded areas are then identified by mismatching disparity values.
- The *ordering constraint* states that the order of neighbouring correspondences on the corresponding epipolar lines has to be preserved. This is used to construct a dynamic programming scheme in order to remove outliers.

The error function is usually only evaluated at discrete pixel steps, yielding pixel-discrete disparity values. To obtain sub-pixel accuracy, a parabola fit on the three points around the minimum is used as a post-processing step [55], as illustrated in the right image of Figure 2.10. The minimum of the parabola gives then the resulting sub-pixel disparity. The motivation for the parabola fit comes from the assumption of a linear image function in the surroundings of the minimum.

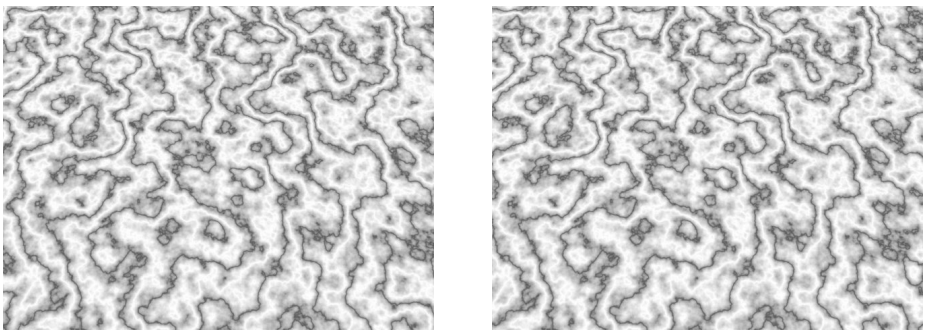

Figure 2.11: Ground truth stereo image pair used to evaluate the distribution of the measured disparities.

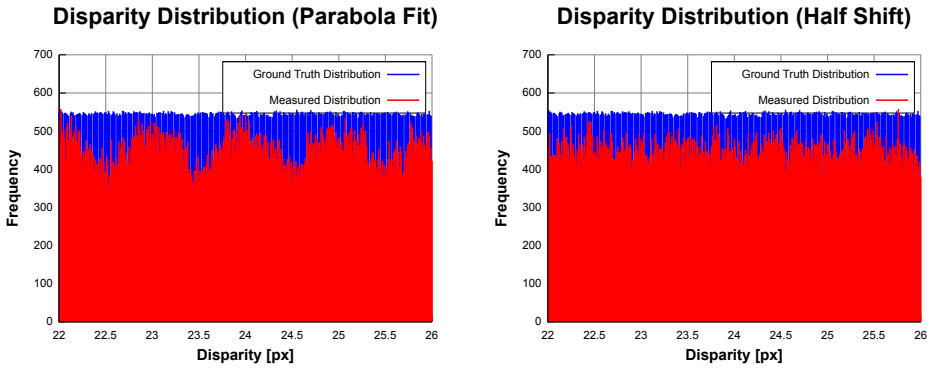

Figure 2.12: Distribution of the measured disparities using the parabola fit and the half shift method.

To analyse the distribution of the disparities, a synthetic stereo image pair is used, as shown in Figure 2.11. Here, a highly structured plane is rotated by 67.5 degrees around the x-axis, so that each line has a different disparity. As it can be seen in the left image of Figure 2.12, the distribution of the disparities obtained by the correlation algorithm and refined by a parabola fit does not match the uniform ground truth distribution. In fact, pixel-discrete disparities have a higher frequency than the disparities between them. This effect is known as the *pixel-locking effect*. Shimizu and Okutomi[56] have studied this effect in detail and derived a function to describe the error in the sub-pixel disparity estimation. They proposed to reevaluate the error function around the minimum by shifting the correlation window to each side by 0.5 px, perform another parabola fit on the newly obtained values and average the results of both parabola fits. The resulting disparity distribution for the same scene is shown in the right image of Figure 2.12. Clearly, this method approximates the uniform distribution more closely than the standard approach.

Stein et al. proposed in [57] a different solution to compensate the pixel-locking effect. Starting with the pixel-discrete disparity map, they perform an iterative refinement of the sub-pixel disparity using a gradient-descent method, similar to the popular Kanade-Lucas-Tomasi tracker that is described in more detail in the next section. Besides the translation of the image window, they also estimate the affine parameters allowing to compensate for the projective deformation of the analysed image patch. This does not only yield an approximate uniform distribution of the disparities, but also allows to refine the disparity by more than one pixel. In Figure 2.13, the disparity distribution is shown using such an iterative approach. In contrast to the parabola fit, the variation in the distribution is much smaller but seems to have a slight half-pixel-locking effect.

The ground truth images used to analyse the disparity distribution can

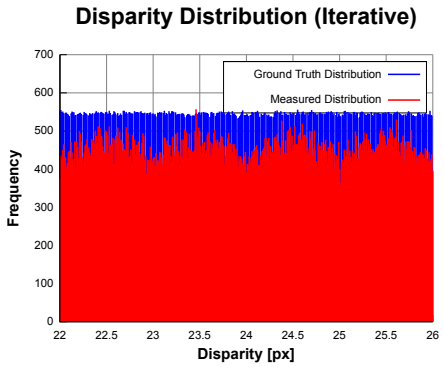

Figure 2.13: Distribution of the measured disparities using an iterative refinement based on the gradient-descent method.

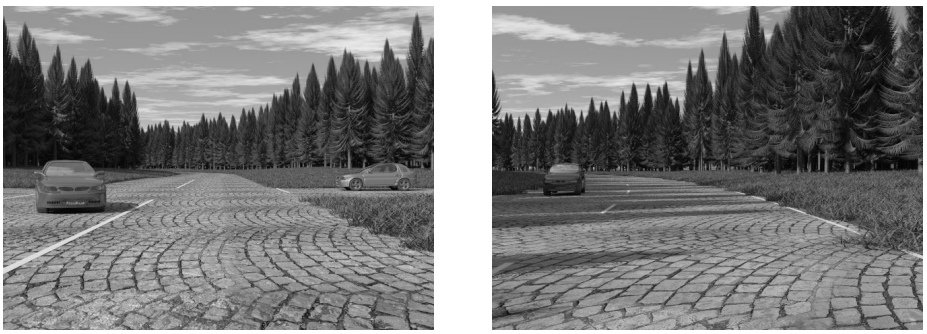

Figure 2.14: Image 223 and 430 of the ground truth sequence used to evaluate the algorithms.

be used to evaluate the accuracy of the stereo algorithm too. However, to obtain more realistic results, the accuracy analysis is performed on another synthetic sequence, showing a typical traffic scenario seen by a moving observer, and introducing effects like transparent windows, illumination changes and repetitive patterns. Two images of this sequence of 681 frames are shown in Figure 2.14. Please note that the grass and trees are actually 3D objects, and not textured planes.

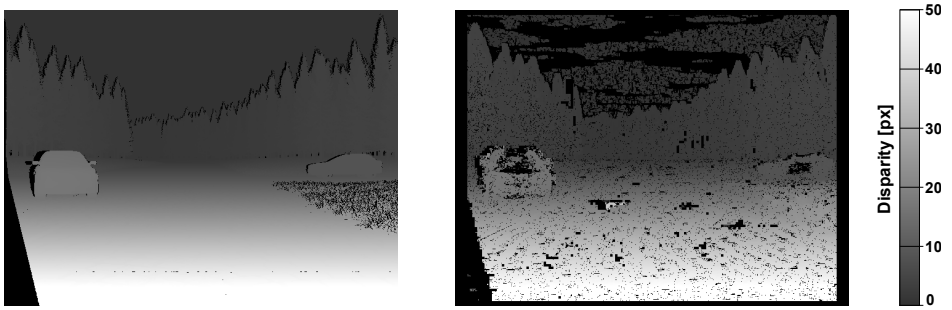

Figure 2.15: Ground truth disparity map and the calculated disparity map of the correlation stereo algorithm.

The ground truth disparity map is shown in the left image of Figure 2.15. The grey value encodes the disparity according to the scale shown on the left side, and occluded pixels are marked black. The right image shows the disparity map obtained by the correlation stereo algorithm. Here, black pixels indicate positions for which no disparity value could be calculated. This happens if the pixel lies in a textureless region, like the homogeneous regions in the sky, or if the disparity was rejected by a verification test during or after the correlation process.

The results of the evaluation on the presented ground truth sequence is shown in Table 2.1 for the parabola fit sub-pixel refinement and in Table 2.2 for the half shift method. Each evaluation is split into an analysis of the errors in non-occluded areas, and in occluded areas where no measurement is possible. In the literature, the root mean square error (RMS) is the most popular error metric, that is defined as

$$\mathrm{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - d_i^*)^2} \tag{2.26}$$

with d_i as the measured and d_i^* as the ground truth disparity. This error can be interpreted as the standard deviation of the errors around the theoretical error of 0 px. However, due to the quadratic penalisation of the error, single outliers can cause a large impact on this metric. Therefore, the average absolute error (AAE) and its standard deviation are presented as well. In

addition, the percentage of errors over 0.5, 0.75, 1.0, 1.5 and 2.0 pixels is given by the values denoted R 0.5, R 0.75, R 1.0, R 1.5 and R 2.0. These values give an impression of the distribution of large errors, and can be interpreted as the probability of outliers given a certain threshold.

	Non-occluded	Occluded
AAE	0.427 px (σ=1.853 px)	3.906 px (σ=8.691 px)
RMS error	1.902 px	9.528 px
R[4]0.5	11.16 %	74.96 %
R 0.75	6.54 %	69.79 %
R 1.0	5.17 %	64.84 %
R 1.5	3.98 %	54.61 %
R 2.0	3.25 %	46.62 %
Coverage	79.95 %	54.29 %
Computation time (CPU)	140 ms	

Table 2.1: Evaluation of the correlation stereo algorithm with a parabola fit for sub-pixel refinement.

	Non-occluded	Occluded
AAE	0.426 px (σ=1.852 px)	3.908 px (σ=8.692 px)
RMS error	1.901 px	9.530 px
R 0.5	11.34 %	75.18 %
R 0.75	6.59 %	69.85 %
R 1.0	5.17 %	64.75 %
R 1.5	3.97 %	54.43 %
R 2.0	3.24 %	46.56 %
Coverage	79.95 %	54.29 %
Computation time (CPU)	257 ms	

Table 2.2: Evaluation of the correlation stereo algorithm using the half shift method for sub-pixel refinement.

Although the method of Shimizu and Okutomi achieved a much better approximation of the uniform disparity distribution, the accumulated errors show only a small improvement with respect to the standard sub-pixel refinement. Taking into account the nearly doubled computation time of the method, here measured on an Intel Quad Core 3.0 GHz system, the additional computational cost clearly does not pay off.

[4]Rx denotes the percentage of disparities with an absolute error larger than x pixel.

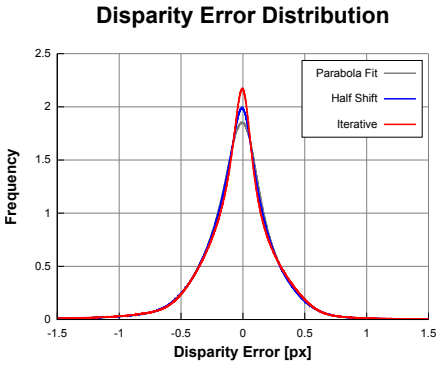

Figure 2.16: Disparity error distribution of the correlation stereo algorithm using a parabola fit (grey), the half shift method (blue) and the iterative method (red) in the sub-pixel refinement step.

The error distribution of the obtained disparity maps for the discussed sub-pixel methods is shown in Figure 2.16. Here, all errors were accumulated over all 681 frames in an histogram and evaluated using robust statistics in order to obtain the inlier distribution. This includes calculating the median of the distribution as a robust estimate of its mean, and the median of the absolute deviations (MAD) to determine the scale parameter of the distribution. The shown distributions are normalised, so that their integral is 1. The distributions of all methods have their maximum at 0 px, which is also the mean of the distributions. The variance of the parabola fit method is $0.049\,\mathrm{px}^2$ ($\sigma = 0.221\,\mathrm{px}$), and the half shift method achieves a slightly lower variance of $0.048\,\mathrm{px}^2$ ($\sigma = 0.219\,\mathrm{px}$). The iterative method performs best in terms of accuracy and yields a variance of $0.045\,\mathrm{px}^2$ ($\sigma = 0.212\,\mathrm{px}$). However, the difference between the individual methods is only small, and again, the higher computational cost of the half shift method, and especially the iterative method, does usually not pay off in real-time applications.

2.3.2 The Kanade-Lucas-Tomasi Tracker

The calculation of correspondences between two successive images, the so called optical flow, is also a main topic in Computer Vision and many different methods are found in the literature. These can be categorised into the following classes [58]:

- *Differential techniques* compute the optical flow from the spatiotemporal derivatives of the image.

- *Region based techniques* estimate the translation of regions in the image by analysing an error function like the SAD, SSD or ZSSD.

- *Energy based techniques* derive the optical flow from an energy analysis of velocity-tuned filters in the Fourier domain.
- *Phase based techniques* retrieve the optical flow from the phase behaviour of band-pass filters applied on the images.
- *Signature based techniques* calculate a unique signature for a region around a centre pixel and derive the optical flow by matching the signatures between the two images.

An extensive introduction to the topic of optical flow computation can be found in most Computer Vision books (e.g. [59, 58]). Here, we will concentrate on the first class, the differential techniques.

From the constant brightness assumption follows for a point projected into the images at time step t and $t + \delta t$:

$$I(\mathbf{x}, t) = I(h(\mathbf{x}, \mathbf{p}), t + \delta t) \tag{2.27}$$

with the image motion function

$$h(\mathbf{x}, \mathbf{p}) = \mathbf{x} + \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} \tag{2.28}$$

modelling a translation. By approximating the image function using the first order Taylor expansion, the intensity at $t + \delta t$ is given by

$$I(h(\mathbf{x}, \mathbf{p}), t + \delta t) \approx I(\mathbf{x}, t) + \frac{\partial I}{\partial u}\delta u + \frac{\partial I}{\partial v}\delta v + \frac{\partial I}{\partial t}\delta t \tag{2.29}$$

leading to the *gradient constraint* equation

$$\nabla I(\mathbf{x}, t)^\top \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} + \frac{\partial I}{\partial t}\delta t = 0 \tag{2.30}$$

with $\nabla I = \left(\frac{\partial I}{\partial u} \; \frac{\partial I}{\partial v}\right)^\top$ as the image gradient.

Equation (2.30) gives a linear relation between the displacement vector and the image function. However, this equation system is under-determined and reveals only the normal component of the image motion. Therefore, in order to estimate the translation vector, additional constraints are necessary. Methods estimating the optical flow field for the whole image usually introduce a smoothness constraint, stating that the optical flow vectors between neighbouring pixels should be approximately the same. A prominent example of such an algorithm is the method reported by Horn and Schunck in [60]. They use the sum of the squared gradients of the flow components as a smoothness term. However, since the deviation of the flow field is penalised quadratically, this algorithm tends to smear the flow field at motion boundaries. To overcome this problem, Nagel proposed to use an *oriented-smoothness* constraint to reduce the effect of smoothing in the presence of

edges [61]. Recently, variational methods were presented that use the L_1 norm to penalise deviations of the smoothness [50, 62], resulting in much clearer motion boundaries. These algorithms usually do not run in real-time on common CPUs, but were recently implemented on modern graphic cards, exploiting the massive parallel architecture of the GPU and achieve frame rates of up to 60 Hz on VGA images.

Another method to introduce a smoothness constraint was described by Lucas and Kanade in [63]. Assuming the image motion is constant over a small image window W, also called *feature window*, they derive the linear equation system from equation (2.30) as

$$\left(\sum_{\mathbf{x} \in W} w\left(\mathbf{x}\right) \, \nabla I\left(\mathbf{x}, t\right)^{\top} \right) \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} = - \sum_{\mathbf{x} \in W} w\left(\mathbf{x}\right) \, \frac{\partial I}{\partial t}\left(\mathbf{x}\right) \delta t \tag{2.31}$$

with $w\left(\mathbf{x}\right)$ as a weight function. The solution is then directly found as

$$\begin{aligned} \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} \; = \; & \begin{bmatrix} \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial u} \frac{\partial I}{\partial u} & \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial u} \frac{\partial I}{\partial v} \\ \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial u} \frac{\partial I}{\partial v} & \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial v} \frac{\partial I}{\partial v} \end{bmatrix}^{-1} \\ & \begin{pmatrix} \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial u} \frac{\partial I}{\partial t} \delta t \\ \sum w\left(\mathbf{x}\right)^2 \, \frac{\partial I}{\partial v} \frac{\partial I}{\partial t} \delta t \end{pmatrix}. \end{aligned} \tag{2.32}$$

Looking at this equation, the optical flow vector can only be derived if the matrix to invert is non-singular. The matrix represents the amount of structure of the image patch, and is also known as the *structure tensor*. In fact, the linear equation system is only well-conditioned, if the structure tensor is well above the noise level. Therefore, Tomasi and Kanade proposed in [64] to estimate the optical flow only for regions with a structure tensor whose eigenvalues are above a given threshold. Since the upper limit of the larger eigenvalue of the structure tensor is defined by the discretization of the grey values, only the smaller eigenvalue has to be analysed.

As the replacement of the image function by its first order Taylor expansion is only justified in the proximity of the solution, the linearization is performed iteratively until convergence. Since the displacement vector is not discretized, the calculation of the structure tensor and grey value difference vector must be performed around a sub-pixel position of the image. Therefore, the grey values of the image are interpolated using bi-linear or cubic interpolation. In order to speed-up the process and allowing to estimate large flow vectors, image pyramids are used: The flow field is first calculated on a sub-sampled version of the original image pair, and successively refined on the levels with higher resolutions.

To establish correspondences over multiple frames, the position of the feature is successively updated by the estimated translation vector. This results in the well-known Kanade-Lucas-Tomasi tracker (KLT), that tracks

a given number of features over the image sequence. New features are established on regions of high structure by the feature detector, that analyses the minimum eigenvalue of the structure tensor, and tracked features are successively updated using the given iterative flow estimation scheme. To verify the consistency of the updated position, different verification techniques have been proposed. The simplest one is to check the sum of absolute differences between the image patch in the last and the current image. If it exceeds a given threshold, the flow and therefore the track is interpreted as a mismatch and the feature is deleted. A better approach would be to verify the similarity between the first appearance of the feature and the current one. However, due to scaling effects, such an approach will not work for points whose distance to the camera changes. Therefore, Shi proposed in [65] to estimate the affine deformation of the feature window between the first and the current frame and use its residuum as an indicator for the similarity. However, this verification strategy is computational expensive and therefore usually not used in real-time applications.

The maximum number of features tracked by the KLT and the maximum number of iterations give an upper bound on the required computational resources, that allows to tune the computation time of the KLT easily. Since the features are tracked independently from each other, this algorithm can be parallelized on the feature-level, and was recently ported to the GPU. Such an implementation was realised by the author and can track up to 10000 features on 4 pyramid levels with a maximum number of 20 iterations per level in less than 20 ms, with a constant time to detect new features of about 5 ms[5].

Evaluating the obtained flow fields using the introduced ground truth sequence leads to the error measures shown in Table 2.3. Here, the average endpoint error (AEE) is defined as the mean euclidean distance between the ground truth and the measured end point as

$$\text{AEE} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(u_i - u_i^*)^2 + (v_i - v_i^*)^2} \tag{2.33}$$

with u_i^*, v_i^* as the components of the ground truth flow vector. The root mean square error (RMS) gives the standard deviation of the error, assuming it has zero mean:

$$\text{RMS}=\sqrt{\frac{1}{n}\sum_{i=1}^{n}(u_i - u_i^*)^2 + (v_i - v_i^*)^2}. \tag{2.34}$$

The average angular error (AAE) of a flow vector was first introduced by Fleet and Jepson in [66], and is defined as the angle in the 3D space between

[5]The implementation is based on the CUDA programming model and the computation time was measured on a NVidia GeForce GTX 285.

the vectors $(u, v, 1)^\top$ and $(u^*, v^*, 1)^\top$. It gives a relative measure of performance and avoids the "divide by zero" problem for zero flows. In practice, errors in small flows are penalised more than errors in large flows, thus giving an unbalanced error measure [67]. However, it is included in the evaluation due to its popularity.

	Non-occluded	Occluded
AEE	0.238 px (σ=0.586 px)	1.064 px (σ=1.401 px)
AAE	3.56° (σ=7.48°)	11.65° (σ=20.20°)
RMS error	0.632 px	1.759 px
R 1°	60.69 %	68.43 %
R 3°	29.38 %	42.69 %
R 5°	17.32 %	33.29 %
R 0.1 px	66.65 %	91.25 %
R 0.5 px	7.53 %	57.97 %
R 1.0 px	2.86 %	34.84 %
Coverage	2.49 %	1.25 %
Computation time (GPU)[6]	17 ms	

Table 2.3: Evaluation of the Kanade-Lucas-Tomasi tracker using a maximum of 10000 features.

In addition, the percentage of measurements exceeding a given error threshold is given in the table, that gives an impression about the distribution of the errors. For the angular error, the values R 1°, R 3° and R 5° denote the percentage of measurements with an angular error *above* 1°, 3° and 5°. For the endpoint error, the thresholds are 0.1 px, 0.5 px and 1.0 px.

To analyse the accuracy of the optical flow estimated using the described Kanade-Lucas-Tomasi tracker, the ground truth sequence shown in Figure 2.14 is used, and the endpoint errors of the flow vectors are accumulated over all 681 frames. Again, robust statistic is used to obtain the inlier distribution. The normalised error distributions are shown in the left image of Figure 2.17. Here, the variance of the error in u is $0.016\,\mathrm{px}^2$ ($\sigma_u = 0.13\,\mathrm{px}$) and in v $0.013\,\mathrm{px}^2$ ($\sigma_v = 0.11\,\mathrm{px}$), with their means at $0\,\mathrm{px}$. The shapes of the error distributions resemble a Laplacian distribution.

Since the KLT tracks features by concatenating the flow vectors, the variance of the endpoint error with respect to the first position of the feature increases linear with the number of tracked frames, according to the laws of classic error propagation. The actual evolution of the variance can be seen in the right image of Figure 2.17. Here, the variance increases ap-

[6]The computation time includes the pyramid preparation and the tracking, but does not include the time to refill the feature pool (about 5ms).

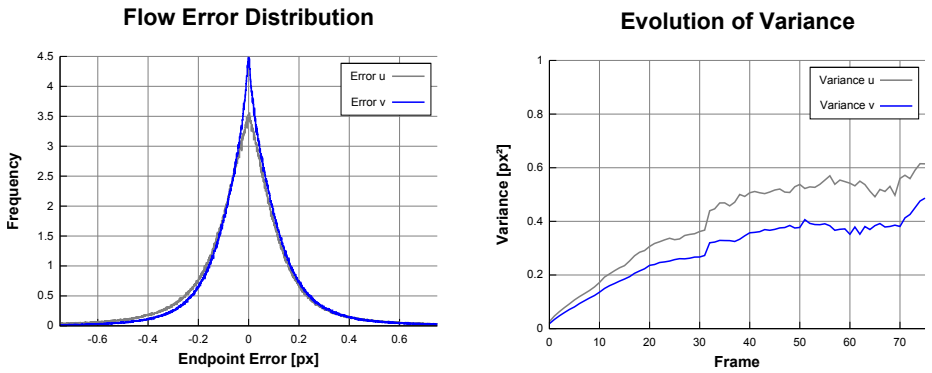

Figure 2.17: Error distribution of the optical flow components u (grey) and v (blue) estimated by the Kanade-Lucas-Tomasi tracker (left), and evolution of the variance over multiple frames (right).

proximately linear over the first frames. However, after about 20 frames, the variance seems to be increasing less. This can be explained by the type of features that were tracked. Features lying on close-by world points have large displacement vectors, and move quickly out of the image, so they contribute only to the first frames. Features lying on points far away from the observer, for example points on the clouds, can be tracked a long time, have only small displacements and nearly no scale effect, resulting in a much lower variance of their flow error.

Chapter 3

The 6D-Vision Principle

In this chapter, the basic principle for three-dimensional motion estimation based on the fusion of stereo information and optical flow is presented. With the knowledge of the 3D reconstructions and the correspondences between two frames, one can easily calculate the three-dimensional motion directly. However, as we will see, this differential combination will lead to unsatisfactory results due to the immanent measurement noise. Hence, instead of combining stereo and optical flow only once for every stereo image pair, the information is integrated over time. This is accomplished using a Kalman Filter that recursively corrects its state estimate and is therefore well-suited for real-time applications. As the system state vector contains six state variables, namely the coordinates of the 3D position and the components of the 3D motion vector, this combination is called *6D-Vision*.

For a real-time system, the speed of convergence of the estimator is crucial to react in time. It is shown, that the speed of convergence highly depends on the initialisation of the state. In general, the motion components of the state are unknown, which is also called the initialisation problem. To overcome this problem, multiple estimators initialised or parametrised differently are used in parallel, and a unified decision strategy decides which estimator performs best.

3.1 Introduction

3.1.1 Track-before-detect Motion Estimation

To determine the 3D motion of independently moving objects in a scene, common methods identify the objects first, and measure their motion afterwards. The objects can be detected by various methods, for example by classification, depth analysis or optical flow segmentation. Once an object is found, it is tracked over two or multiple frames and its motion parameters are derived. However, such a *track-after-detect* strategy highly relies on

the quality of the detection step, that is usually tuned to have a low false negative rate.

Track-before-detect methods reverse both steps and are known to provide much better results in situations of high input noise [68, 69]. Here, object hypothesis are tracked over time and the following detection step benefits from the reduced noise levels of the temporal integration. The tracking step can also estimate hidden model parameters, like for example the velocity, that increases the robustness of the detection step even further.

The 6D-Vision algorithm allows to realise such track-before-detect systems, as it estimates the 3D motion of a large number of image points independently, using no object knowledge at all. Independently moving objects are then easily detected in the 3D motion field as clouds of 3D points with coherent motion vectors. This reveals not only the approximate 3D boundaries of the object, but also the objects linear 3D motion as it is already estimated by the algorithm.

3.1.2 The Need for Filtering

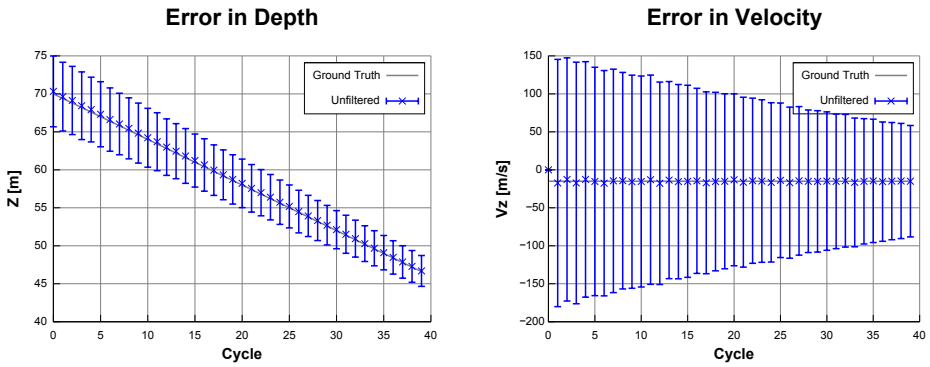

Figure 3.1: Mean depth (left) and mean differential depth velocity (right) of a moving point. The error bars show the σ-range of all samples.

A naive approach to derive the 3D velocity of points seen by a stereo camera system would use the change of two successive 3D positions directly. Such an differential approach would use the optical flow to determine the correspondence of the image point between two frames, and a stereo algorithm to determine its disparity and therefore its 3D position. Due to the immanent measurement noise, the 3D position is only known approximately, and especially distant points suffer from a high depth uncertainty.

Take a look at Figure 3.1. Here, a moving point was simulated starting at the world position $(2.0\,\mathrm{m},\ 1.0\,\mathrm{m},\ 70.0\ \mathrm{m})^\top$ and moving at a constant velocity of $(2.0\,\mathrm{m/s},\ 0.1\,\mathrm{m/s},\ -15.0\,\mathrm{m/s})^\top$. The image measurements were taken at intervals of $\Delta t = 0.040\,\mathrm{s}$ and disturbed by a white Gaussian noise with the

variances $\sigma_u^2 = \sigma_v^2 = 0.01\,\text{px}^2$ and $\sigma_d^2 = 0.05\,\text{px}^2$. The stereo camera system has a focal length of $f_u = f_v = 800.0\,\text{px}$ and a base width of $b = 0.30\,\text{m}$. The experiment was repeated 10000 times and the mean depth and the σ_z range are shown in the left image of Figure 3.1. The mean depth is slightly overestimated at the beginning, due to the stereo bias discussed in Section 2.2.3. The mean differential velocity and its σ_{v_z} range is shown in the right image of the same Figure. Although the mean is in the proximity of the ground truth value, the high variance of the derived depth velocity makes an application to detect moving objects impossible. In order to obtain a valid velocity, it is therefore necessary to integrate the measurements over time and incorporate their uncertainties.

To determine the hidden state of a system from a number of observations, a model of the underlying physical process, the so called *system model*, and the relation between the observations and the state, the *measurement model*, are required. Both models have to cope with uncertainties: The measurement model has to cope with the immanent measurement noise, whereas the uncertainty of the system model is due to the approximation of the real physical process. Therefore, the determination of the state becomes an estimation process.

In 1960, Rudolph E. Kalman presented the Kalman Filter to solve the estimation problem for a time-discrete linear dynamic system in an elegant manner [70]. The state is estimated from a series of noisy measurements recursively, so that at each time step an estimate is available, without the need of reprocessing all previous measurements. The Kalman Filter is often referred to as an optimal estimator, in the sense that the variance of the estimated state is minimised. Although designed for linear system and measurement models, it is widely used to estimate non-linear systems, which leads to the so called *Extended Kalman Filter* [71, 72]. A detailed discussion of the Kalman Filter and the Extended Kalman Filter is found in [73, 74, 75]. A comprehensive collection of the original publications on Kalman Filters is available in [72].

3.1.3 Related Work

Normal flow reflects only the motion in the direction of the image gradient, but has some advantages with respect to optical flow in the sense that it reduces the correspondence problem. In [76], Argyros and Orphanoudakis proposed a method based on normal flow fields and the least median squares, which estimates simultaneously the ego-motion and independent 3D motion. Morency and Darrell [77] have also proposed a method for pose-estimation based on normal flow and the Iterative Closest Point algorithm [78].

Methods based on the optical flow have been widely investigated. One of the first attempts to fuse stereo and optical flow information was studied by Waxman and Duncan in [79], exploiting the relationship between

the 3D motion and image velocities with stereo constraints. Kellman and Kaiser [80], Mills [81] and Heinrich [82] also use such geometric constraints to detect independent motion. Demirdjian and Horaud proposed in [83] a method for the estimation of the ego-motion and the segmentation of moving objects. Demirdjian and Darrel [84] estimate rigid motion transformations by mapping two reconstructions of a rigid scene in the disparity space.

Kalman Filters for object tracking are used widely in the computer vision literature. Some of the most significant methods are shortly described here.

Dang et al. [85] fuse the optical flow and the stereo disparity using Kalman Filters for object tracking. Based on an existing detection and segmentation of the object, an Extended Kalman Filter estimates the initial 3D positions of the tracked image points, the 3D velocity of the object and its accumulated 3D translation vector from the initial to the current position. A test based on the Mahalanobis distance is performed in order to eliminate those points with incoherent motion and which possibly do not belong to the object being represented by the remaining observations.

Suppes et al. [86] estimate landmark positions obtained with stereo and filter them over time with Kalman Filters. The projection of the PDF of the points on a depth map allows the accurate detection of stationary obstacles. Phantom objects are also less probable to appear this way, since the lifetime of a false correspondence is normally very short and, therefore, its covariance matrix is large. The required ego-motion is obtained from the inertial sensors of the robot.

Sibley et al. [42] use Kalman Filter for modelling the dynamic of distant points measured with stereo. The dynamic of the point is left unspecified and is assumed to be given. An analysis of the bias in the triangulated 3D points is carried out and a correction using a second order approximation of the triangulation function is proposed.

Lee and Kay [87] estimate object motion using Kalman Filters in stereo image sequences. The position and orientation as well as the translational and rotational velocities of the object are estimated. The authors present first the camera geometry and derive simplified linear expression relating measurement noise of a feature point in a stereo image and its position error induced in 3D space. A differential rotation matrix is defined and a least squares expression is found for its computation. A measurement equation for the Kalman Filter is found using quaternions to represent rotation and differential rotation.

Rives et al. [88] present one of the first structure from motion algorithms using Kalman Filters and normal flow using monocular images. The authors derive the equations of the image velocity field given the motion of the camera, and then eliminate the rotation from the equations in order to simplify the analysis. A solution for the depth of the tracked point given the velocity field is obtained. The observers translation is refined by minimising a cost function relating the normal flow and the motion parameters from

the inertial sensors.

Matthies et al. [89] propose an iconic disparity map estimation using Kalman Filter with a purely lateral translational monocular camera. An analysis of the accuracy of the estimated distance of 3D points according to the direction of the camera motion is carried out. As a result the authors obtain the relative precision of stereo and depth from motion. The authors point out the importance of taking into account the off-diagonal elements of the state covariance matrix in order to model smoothness in the disparity map. In addition, a feature based model is presented and compared with the iconic one. The feature based approach has a quicker convergence rate because it keeps the disparity and the sub-pixel-position of the feature as state elements while the iconic only keeps the disparity. A comparison with stereo shows the interesting result that processing the intermediate frames (as the camera moves laterally) does not improve the precision if compared to computing stereo on the first and last frames of the images sequence.

Zhang and Faugeras [90] present a complete framework for the segmentation of objects and the computation of their 3D motion using a trinocular stereo camera system. Correspondences for image lines are found in space and time (stereo and optical flow). The line is accordingly represented with a mid-point and a direction vector so that the full covariance matrix is considered. Kalman filters are used to estimate angular velocity, translational velocity and translational acceleration of the detected objects. The Mahalanobis distance between predicted line segment and measured line segments is used to select possible matches. In addition, a bucketing technique is used to reduce the number of hypotheses even more. All remaining hypothesis are then tracked in order to observe their dynamic. The mahalanobis distance is once again used to eliminate features tracked wrongly. The grouping of tokens into objects is performed based on the Mahalanobis distance between the motions of two tokens. The covariance matrix of the object is computed and used to iteratively check if other tokens belong to the object.

Altunbasak et al. [91] estimate 3D point motion with a maximum likelihood approach and Kalman Filters in stereo sequences. Kalman Filters are used to model point position, translation velocity, translation acceleration, rotation, angular velocity and precession. Stereo and motion are fused in this way by maximising the probability that the estimated motion and disparity conform to the observed frames. The conditional probability distribution is modelled as a Gibbsian distribution. The algorithm then iterates between the maximum likelihood step and the Kalman Filter step until the maximum likelihood cost can no longer be reduced.

Yao and Chellappa [92] present a method for tracking features using Kalman Filters in an image sequence. The state model considers image position, image velocity and rotation of the features. The Mahalanobis distance is used to choose potential features points in the next image. The zero-mean normalised cross-correlation function is used for matching feature

points. A new feature point is added for tracking if the probability of this feature to be a feature in the previous image is very small.

Hung and Tang et al. detect and track multiple moving objects computing stereo and optical flow in left and right images in [93] and [94]. A mutually-supported consistency constraint is used to reduce errors in the feature matching process. RANSAC[1] is then used to find clusters of points with similar motion, where similarity is defined as the inverse Euclidean distance between the point position and the predicted position. Kalman Filters are used on each detected cluster in order to track every object in the scene. Angular velocity, angular acceleration, point of rotation, translational velocity and translational acceleration build up the state vector. When tracking a feature, the prediction for the corresponding cluster is used to predict the 2D image position and support the correspondence this way.

Kalman Filters are widely used in self localisation and mapping (SLAM) applications. Jung and Lacroix [96], for example, describe a method for building digital elevation maps using stereo images. The Kalman Filter is used to simultaneously refine estimates of ego-motion and 3D landmark position of world points. Only a sub-set of the dense output provided by the stereo algorithm are used as landmarks for the computation of the ego-motion. The rest of the stereo output is used to build maps of the environment. The state vector includes in this way the six motion parameters of the camera and the 3D position of every tracked point.

Matthies and Shafer [97] also estimate landmark positions in a camera-centred coordinate system using Kalman Filters. The ego-motion is computed from the 3D points obtained with stereo. The covariance matrix of each stereo point is used to compute a motion covariance matrix, which is then propagated to the covariance matrices of the landmarks. The update of the global robot position is carried out by concatenating the transformation matrices and estimating the uncertainty of the global position by propagating the covariances matrices of the incremental motions into a covariance of the global position.

Kalman Filters are not the only tool for combining stereo and motion components. Some of these methods are shortly described in the following.

Liu and Skerjanc [98] present a method for finding stereo and motion correspondences using a coarse-to-fine strategy. Dynamic programming is used with a cost function including interline penalty, a motion penalty and pyramid penalty components. Dynamic programming is applied on every level of the pyramid. The authors also give some geometric relationships between motion and disparity.

Jenkin and Tsotsos [99] present a method for handling multiple matching hypothesis generated from a stereo image sequence. They describe some

[1] RANdom SAmple Consensus: An iterative method to estimate the parameters of a model from a set of observations, presented by Fischler and Bolles in [95].

smoothness assumptions and define constraints based on such assumptions. Features are tracked in 3D and multiple hypotheses are generated in a tree-like structure. To every node a label is assigned and some combinations of labels are found to be incoherent, i.e., a false correspondence.

Ho and Pong [100] present a method for matching features in two consecutive stereo images. Four matchers are integrated as a network. In the first step, features are extracted from the images. Then, multiple hypothesis are established for every feature, and to each potential match an initial probability is assigned. In the last step, the probabilities are updated iteratively by a relaxation labelling process.

Altunbasak et al. propose in [101] a framework for the simultaneous motion and disparity estimation including scene segmentation. They present a Bayesian framework to estimate the motion with 6 degrees of freedom, with the probabilities distribution modelled as Gibbs distributions. The algorithm iterates between computing the maximum a posteriori estimate of the disparity and the segmentation fields, conditioned on the present motion parameter estimates and the maximum likelihood estimates of the motion parameters via simulated annealing.

Agrawal et al. [102] present also a complete framework for detecting independently moving objects. After calculating the ego-motion, the previous image is warped to the current time step according to the ego-motion. Independently moving objects are identified as blobs in the difference image between the warped and the current one, and tracked over time. Multiple hypothesis of motion are generated using RANSAC. To every hypothesis a score is assigned depending on the error of the projection of the points based on the current motion hypothesis. The motion hypothesis with a higher score is then used as the starting point for a non-linear minimisation using the Levenberg-Marquardt algorithm. The function to minimise is the projection error. The authors use the method of Demirdjian [84] to compute this projection error.

Talukder and Matthies [103] present a similar method for the detection and tracking of independent moving objects. Independent motion is found by first computing the ego-motion of the camera with respect to the static scene, and then observing the difference between the predicted and the measured optical flow and disparity fields. These differences are thresholded in order to build a binary map. Moving objects are then detected in this map as binary blobs. The segmentation of moving objects is performed with a simple algorithm based mainly on heuristics. The method requires dense optical flow and disparity fields.

3.2 Kalman Filter based Motion Estimation

3.2.1 The System Model

Let us describe a moving world point by the six-dimensional state vector $\mathbf{x} = (x, y, z, \dot{x}, \dot{y}, \dot{z})^\top$ consisting of its three-dimensional position and its three-dimensional velocity vector. The time-continuous system model in a fixed world-coordinate system is then given by the differential equation system

$$\dot{\mathbf{x}}_w(t) = \mathbf{A}_w \mathbf{x}_w(t) + \boldsymbol{\omega}_w(t) \tag{3.1}$$

$$= \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \mathbf{x}_w(t) + \boldsymbol{\omega}_w(t) \tag{3.2}$$

where $\{\boldsymbol{\omega}_w(t)\}$ is a continuous-time white noise process, so its mean vector is $\mathbf{0}_6$ and its covariance matrix is $\mathbf{Q}_w$. This covariance matrix represents the uncertainty of the system model, resulting from the approximation of the physical process.

The state transition matrix $\mathbf{A}_{k|w}$ of the corresponding time-discrete system model

$$\mathbf{x}_{k|w} = \mathbf{A}_{k|w} \mathbf{x}_{k-1|w} + \boldsymbol{\omega}_{k|w} \tag{3.3}$$

is found by using the matrix exponential function (a derivation is found in Appendix A). This leads to

$$\mathbf{A}_{k|w} = e^{\mathbf{A}_w \Delta t} = \sum_{i=0}^{\infty} \frac{\mathbf{A}_w^i \, \Delta t^i}{i!} \tag{3.4}$$

$$= \begin{bmatrix} \mathbf{I}_3 & \Delta t \, \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 \end{bmatrix} \tag{3.5}$$

with Δt as the time interval between the steps $k-1$ and k.

The covariance matrix $\mathbf{Q}_{k|w}$ of the time-discrete system model is derived from the time-continuous covariance matrix

$$\mathbf{Q}_w = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{Q}_{21} & \mathbf{Q}_{22} \end{bmatrix}, \qquad \mathbf{Q}_{12} = \mathbf{Q}_{21}^\top \tag{3.6}$$

with $\mathbf{Q}_{i,j}$ as a 3×3 sub-matrix of the covariance matrix $\mathbf{Q}_w$, as

$$\mathbf{Q}_{k|w} = \int_0^{\Delta t} e^{\mathbf{A}_w \tau} \mathbf{Q}_w e^{\mathbf{A}_w \tau^\top} d\tau \tag{3.7}$$

$$= \begin{bmatrix} \mathfrak{Q}_{11} & \mathfrak{Q}_{12} \\ \mathfrak{Q}_{21} & \mathfrak{Q}_{22} \end{bmatrix} \tag{3.8}$$

with the elements

$$\mathfrak{Q}_{11} = \Delta t\,\mathbf{Q}_{11} + \frac{1}{2}\Delta t^2\,(\mathbf{Q}_{12} + \mathbf{Q}_{21}) + \frac{1}{3}\Delta t^3\,\mathbf{Q}_{22} \tag{3.9}$$

$$\mathfrak{Q}_{12} = \Delta t\,\mathbf{Q}_{12} + \frac{1}{2}\Delta t^2\,\mathbf{Q}_{22} \tag{3.10}$$

$$\mathfrak{Q}_{21} = \Delta t\,\mathbf{Q}_{21} + \frac{1}{2}\Delta t^2\,\mathbf{Q}_{22} \tag{3.11}$$

$$\mathfrak{Q}_{22} = \Delta t\,\mathbf{Q}_{22}. \tag{3.12}$$

This system model describes the linear movement of a world point. Although the system model can be easily extended to account for higher-order terms, like for example the acceleration, the presented Kalman Filter is limited to a linear motion for two reasons: First, the uncertainty of such higher-order terms estimated from successive point measurements decreases only slowly, so that it takes a long time to obtain reasonable estimation results. This would be contradictory to the objective of yielding prompt results. Secondly, it increases the computational complexity and therefore the computation time significantly.

The approximation of the underlying physical process by neglecting the higher-order terms of the motion results in a uncertainty of the velocity components. As described in Appendix A, the covariance matrix of the continuous-time system gives the (co)variances over a given time interval. Assuming further, that the uncertainties of the velocity components are not correlated, the covariance matrix $\mathbf{Q}_w$ can be written as

$$\mathbf{Q}_w = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{1}{\Delta t}\mathrm{diag}\left(\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right) \end{bmatrix} \tag{3.13}$$

and the corresponding covariance matrix of the discrete-time process is then given by

$$\mathbf{Q}_{k|w} = \begin{bmatrix} \frac{1}{3}\Delta t^2\,\mathrm{diag}\left(\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right) & \frac{1}{2}\Delta t\,\mathrm{diag}\left(\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right) \\ \frac{1}{2}\Delta t\,\mathrm{diag}\left(\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right) & \mathrm{diag}\left(\sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{z}}^2\right) \end{bmatrix}. \tag{3.14}$$

Up to now, the linear movement of the world point is described in a fixed world coordinate system. Keeping the following measurement and analysis steps in mind, it is more suitable to describe the world point in the observers coordinate system at each time step. Between two time steps $k-1$ and k, the observer coordinate system undergoes a motion that is described by a rotation matrix $\mathbf{R}_k$ and a translation vector $\mathbf{t}_k$, so that points described in the observer coordinate system at time step $k-1$ are transformed to the coordinate system at time step k by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_k = \mathbf{R}_k \begin{pmatrix} x \\ y \\ z \end{pmatrix}_{k-1} + \mathbf{t}_k \tag{3.15}$$

and the associated velocity vector is transformed by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_k = \mathbf{R}_k \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix}_{k-1} . \tag{3.16}$$

This leads to the following discrete-time system model to describe the linear motion of a world point in the coordinate system of the moving observer:

$$\mathbf{x}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{u}_k + \boldsymbol{\omega}_k \tag{3.17}$$

with

$$\mathbf{A}_k = \mathbf{D}_k \mathbf{A}_{k|w} \tag{3.18}$$

$$\mathbf{D}_k = \begin{bmatrix} \mathbf{R}_k & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{R}_k \end{bmatrix} \tag{3.19}$$

$$\mathbf{u}_k = \begin{pmatrix} \mathbf{t}_k \\ \mathbf{0}_3 \end{pmatrix} \tag{3.20}$$

$$\boldsymbol{\omega}_k \sim \mathcal{N}(\mathbf{0}_6, \mathbf{Q}_k) \tag{3.21}$$

The motion of the observer is measured by sensors or estimated from the image sequence, resulting in an uncertainty of the components $\mathbf{R}_k$ and $\mathbf{t}_k$. Depending on the actual calculation, the relation between the measured entities and the derived rotation matrix and translation vector may be non-linear, and the noise of both derived components may be correlated. With the p-dimensional parameter vector $\mathbf{p}_k$ of the observers motion, the rotational and translational components are

$$\mathbf{R}_k = \mathbf{R}(\mathbf{p}_k + \boldsymbol{\varsigma}_k) \tag{3.22}$$

$$\mathbf{t}_k = \mathbf{t}(\mathbf{p}_k + \boldsymbol{\varsigma}_k) \tag{3.23}$$

with the additive white Gaussian noise vector $\boldsymbol{\varsigma}_k$ with known covariance $\mathbf{C}_k$. The covariance matrix $\mathbf{Q}_k$ of the discrete-time system model is then

$$\mathbf{Q}_k = \mathbf{D}_k \mathbf{Q}_{k|w} \mathbf{D}_k^\top + \mathbf{J}_k \mathbf{C}_k \mathbf{J}_k^\top \tag{3.24}$$

with $\mathbf{J}_k$ as the Jacobian

$$\mathbf{J}_{[i,j]} = \left. \frac{\partial \mathbf{x}_{[i]}}{\partial \mathbf{p}_{[j]}} \right|_{(\mathbf{x}_{k-1}, \mathbf{p}_k)} \qquad i = 1 \ldots n,\ j = 1 \ldots p. \tag{3.25}$$

Please note, that if the relation between the parameter vector and the motion components is non-linear, equation (3.25) gives only an approximate solution for the covariance propagation, since higher-order terms are neglected.

3.2.2 The Measurement Model

At each time step k, the world point described by the system model is projected onto the image point $(u, v, d, 1)^\top$ of a standard stereo camera system. The location of this image point $(u, v)^\top$ is measured using a feature tracking algorithm, whereas the disparity d is measured by a stereo algorithm. The projection of the point $(x, y, z, 1)^\top$ given in the current observer coordinate system is described by equation (2.11) as

$$w \begin{pmatrix} u \\ v \\ d \\ 1 \end{pmatrix} = \acute{\mathbf{\Pi}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.26}$$

with $\acute{\mathbf{\Pi}}$ as the extended projection matrix. To retrieve the euclidean coordinates $(u, v, d)^\top$, each component must be divided by the homogeneous component w, and the resulting projection equation becomes non-linear:

$$\begin{pmatrix} u \\ v \\ d \end{pmatrix} = [\mathbf{I}_3|\mathbf{0}_3] \; \frac{1}{w} \acute{\mathbf{\Pi}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.27}$$

$$= h(\mathbf{x}) \tag{3.28}$$

This leads to the non-linear measurement model

$$\mathbf{z}_k = h(\mathbf{x}_k, \boldsymbol{\nu}_k) \tag{3.29}$$

with $\mathbf{z}_k = (u, v, d)^\top$ as the measurement, h as the non-linear measurement function and $\boldsymbol{\nu}_k$ as the additive measurement noise vector with the known covariance matrix $\mathbf{T}_k$. The measurement function projects the world point components $(x, y, z)^\top$ of the state vector $\mathbf{x}_k$ and contains no relation between the velocity components and the measurement vector. To apply the Extended Kalman Filter, the Jacobian

$$\mathbf{H}_{[i,j]} = \left. \frac{\partial h_{[i]}}{\partial \mathbf{x}_{[j]}} \right|_{(\widehat{\mathbf{x}}_k, \mathbf{0}_3)} \qquad i = 1 \ldots 3, \; j = 1 \ldots 6 \tag{3.30}$$

has to be calculated. As the measurement noise is not transformed, the transformation matrix $\mathbf{V}_k$ is $\mathbf{I}_3$ for every time step k.

3.2.3 The Benefit of Filtering

Figure 3.2 shows the estimation result of the presented Extended Kalman Filter, using the same input data as the differential approach of Figure 3.1. Here, the initial variances of the velocity components were set to $1000\,\mathrm{m}^2/\mathrm{s}^2$

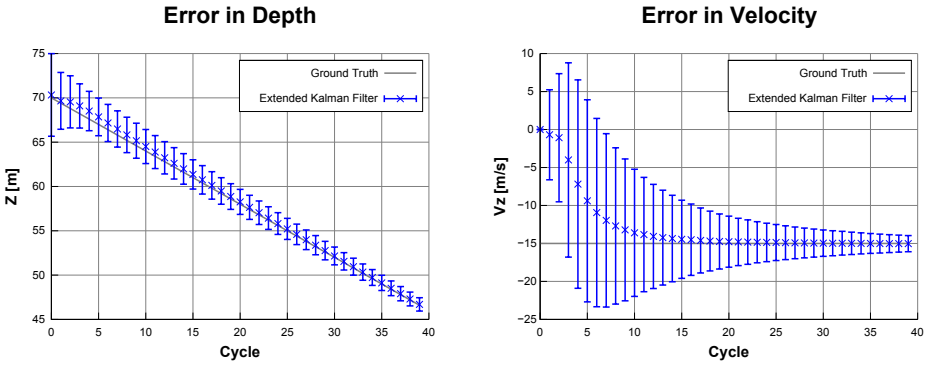

Figure 3.2: Mean depth (left) and depth velocity (right) of the proposed Extended Kalman Filter. The error bars show the σ range of all samples.

and their system variances were set to $0.1\,\mathrm{m}^2/\mathrm{s}^2$. Again, the average of the estimates over 10000 runs and their σ ranges are shown. Obviously, the filter over-estimates the distance of the point over the first frames. This is caused by the initialisation of the filter: At the first cycle, the 3D position reconstructed from the first measurement is used to initialise the position elements in the state. Since no information about the velocity is available at this time, the velocity components are initialised to $0\,\mathrm{m/s}$. As the filter adapts its velocity estimate and converges towards the ground truth velocity, also its depth estimate converges quickly towards the ground truth value.

Comparing the distribution of the filter estimates with the differential approach of Figure 3.1, the benefit of the filtering becomes quite clear. The uncertainty of the velocity estimate decreases drastically with each new cycle, and is much lower than the differential approach. In addition, the uncertainty of the depth estimate is well below the uncertainty of the differential approach.

The Kalman Filter estimates not only the state vector, but also its variance. In Figure 3.3 the variance of the state of 10000 samples was calculated and shown in the grey curve, and the corresponding averaged variance reported by the Kalman Filter is shown in the blue curve. The variance of the depth estimation is shown in the left image. Here, the variance estimated by the Kalman Filter resembles the measured variance very well. In the left image, the variance of the depth velocity is given. Starting at a variance of $1000\,\mathrm{m}^2/\mathrm{s}^2$ set in the initialisation, the estimated variance decreases exponentially. After about 25 frames, the estimated variance decreases slower than the actual state variance. This is not surprising, since the Kalman Filter receives no direct measurement for the velocity and can reduce the state variance only in the shown exponential way.

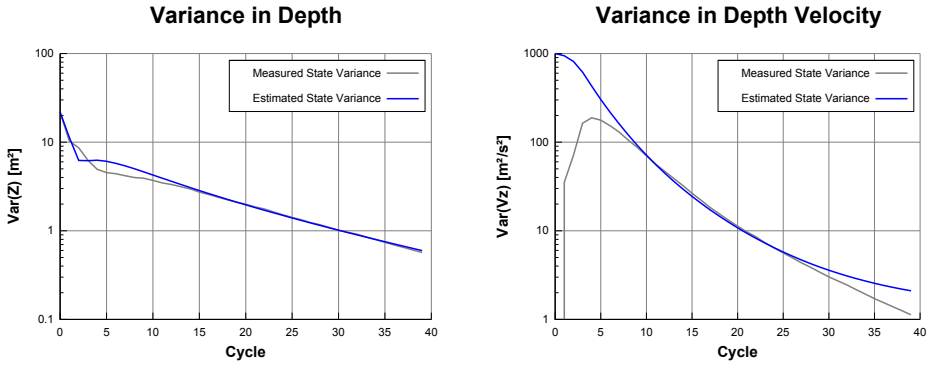

Figure 3.3: Measured and estimated state variance of the proposed Extended Kalman Filter for the depth (left) and the depth velocity (right).

3.2.4 Mastering the Non-Linearity

To account for the non-linear measurement equation, the measurement system was linearised around the predicted state to allow the application of the Kalman Filter. Due to its simplicity, this Extended Kalman Filter is widely used for non-linear estimation problems. However, this approach can fail for wrong initialised filters, especially when the relation between the state and the measurements is highly non-linear. Also, the Extended Kalman Filter is known to under-estimate the state variances in the presence of such situations, and the state estimate and its covariance become inconsistent [104]. This happens if the linearization point suggests a smaller influence of the measurement on the state, and therefore the measurement weights higher, than a correct linearization point would.

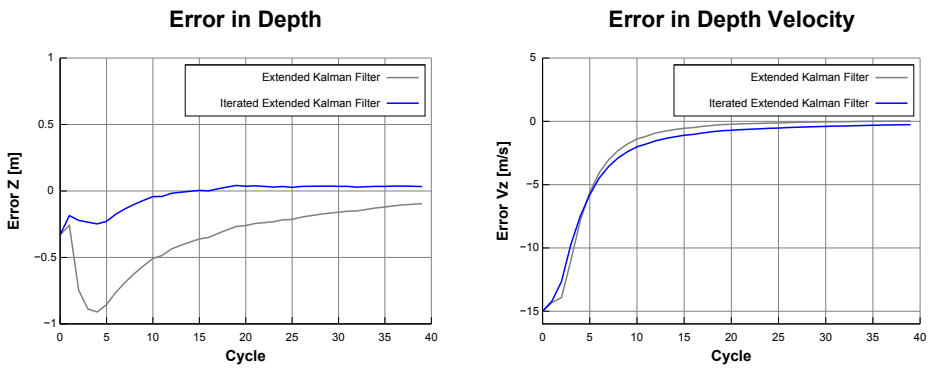

Figure 3.4: Error in the average state estimates for the depth and the depth velocity for the Extended Kalman Filter and the Iterated Extended Kalman Filter.

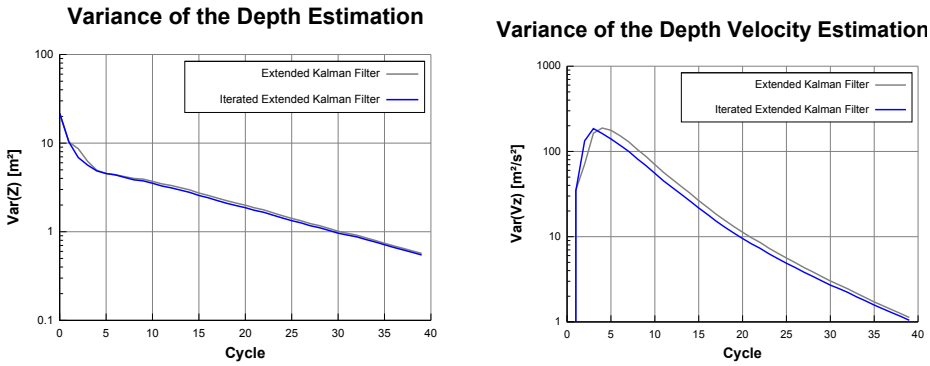

Figure 3.5: Variance of the average state estimates for the depth and the depth velocity for the Extended Kalman Filter and the Iterated Extended Kalman Filter.

One way to overcome this problem is to refine the linearization point iteratively in the measurement update, known as the Iterated Extended Kalman Filter (IEKF). The basic idea is that the updated state gives a better linearization point than the predicted state, and the measurement update is performed repeatedly until the change in the resulting state vector is minimal. In Figure 3.4 the errors of the estimated depth and depth velocity are shown for the discussed Extended Kalman Filter, and the Iterated Extended Kalman Filter using 10 iterations. Obviously, the depth estimate of the IEKF converges much faster to the ground truth depth than the EKF. This was also reported by Sibley in [42], who used an IEKF to estimate the position of static points from stereo measurements. Interestingly, the depth velocity of the IEKF converges over the first 5 frames faster, and then slightly slower than the EKF. Looking at Figure 3.5, the IEKF has overall slightly smaller variances than the EKF. Of course, the better approximation by the IEKF comes at the price of higher computational costs, since the measurement update step has to be repeated n times, whereas the EKF uses only one iteration.

Another way to eliminate the problem of the non-linear measurement model is to estimate not the depth, but the inverse depth. However, this modification introduces a non-linear system model, as the relation between the inverse depth and the depth velocity has to be modelled. In Figure 3.6, the errors of the EKF and such a filter are shown. The error in the resulting depth estimate is for the first 25 frames larger than the error of the EKF, and under-estimates the depth after the frame 35. Looking at the estimated velocity, the drawback of the non-linear system model can be clearly seen. Here, the velocity estimate converges faster than the EKF for the first 15 frames, but again under-estimates the ground truth value.

Clearly, the Iterated Extended Kalman Filter performs best, but requires additional iterations compared to the Extended Kalman Filter. Depending

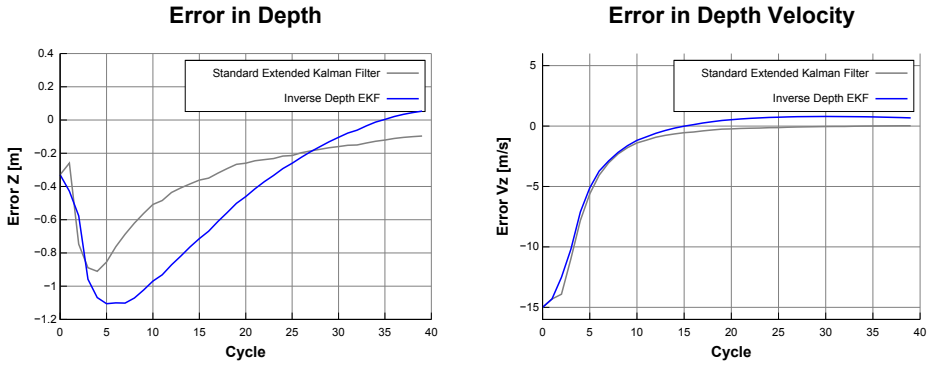

Figure 3.6: Error in the average state estimates for the depth and the depth velocity for the presented Extended Kalman Filter and a modified version estimating the inverse depth.

on the application, and the needs of accuracy, the numbers of iterations may be varied. Also, the number of iterations can be varied according to the estimation cycle, allowing more iterations in the beginning, and less iterations for a steady state.

3.2.5 The Influence of Outliers

Like most estimation methods, the Kalman Filter is known to be sensitive to outliers. It has no build-in method to detect them, and therefore outliers must be identified before fed into the Kalman Filter. Failing to do so can lead to a serious estimation error, especially when an outlier occurs during or shortly after the initialisation, when the measurements have a high impact on the state. In the case of non-linear models, this can also lead to errors in the state covariance estimation, since the influence of the measurement is derived from the linearization around the updated respifnextchar.. predicted state.

To distinguish between inliers and outliers, the deviation between the average measurement and the individual measurements is usually analysed. Robust methods use the median instead of the mean value to obtain a reference value. The outlier detection is often performed by defining a confidence interval around the reference value, and outliers are found as values outside this interval. Such an approach is often used for least-squares methods, which estimate the parameters from a series of recorded measurements.

However, since the Kalman Filter updates its state recursively with each new measurement, the outlier detection can not analyse the evolution of the measurements, as outliers may have already been fed into the Kalman Filter. Therefore, the discrepancy between the predicted measurement and the actual measurement is used instead. This difference is known as the *innova-*

tion vector $\mathbf{s}$ and its covariance matrix is denoted $\mathbf{S}$. From the assumption, that more than 99.7 % of all measurements lie within a range of $\pm 3\,\sigma$ around the predicted state, follows with the Mahalanobis distance

$$\epsilon^2 = \mathbf{s}^\top \mathbf{S}^{-1} \mathbf{s} \tag{3.31}$$

the condition for an inlier $\epsilon \leq 3$, and an outlier is detected as a violation of this constraint. This test also called the $3\,\sigma$-test. Once an outlier is detected, the measurement update step is either skipped and the filter resumes its operation with the next measurement, or the filter is reset.

Please note, that this test can not detect outliers during the initialisation phase, since the large initial state covariance results in a wide acceptance interval. However, as soon as more measurements are incorporated into the state estimate, the acceptance interval shrinks, and wrongly initialised filters are detected and eliminated. Since we assume outliers to occur randomly, the following processing steps must be aware of the possibility of single wrong estimates, but can easily compensate this by using the estimation result of a sufficiently high number of filter estimates.

3.2.6 Limitations

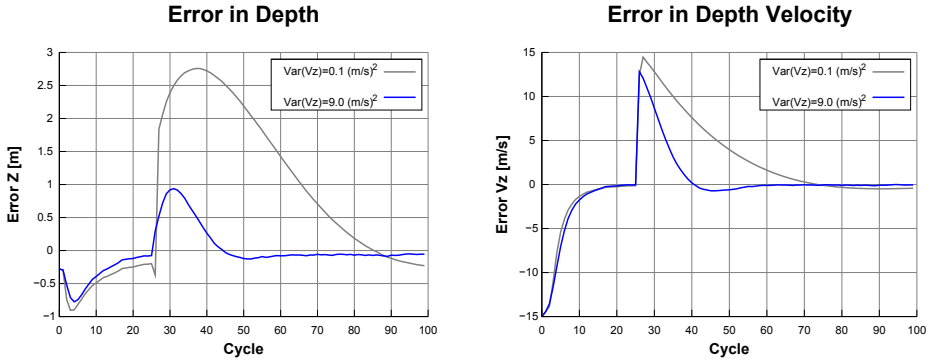

Figure 3.7: Step response of the Extended Kalman Filter for a moving point. The movement stops suddenly at cycle 25.

The system model of the presented Kalman Filter was developed for a point moving at constant speed, with the acceleration modelled by a zero-mean white noise. If the observed point actually undergoes an acceleration, this assumption is violated and the filter has to constantly adapt its velocity estimate. The speed of adaption is limited by the assumed variance of the velocity modelled in the system covariance matrix. This effect can be seen in Figure 3.7. Here the moving point of the previous examples moves at a constant speed $v_z = -15\,\mathrm{m/s}$ and stops moving at cycle 25. The two curves show the same filter parametrised with a velocity variance of $0.1\,\mathrm{m^2/s^2}$ (grey)

and 9.0 m^2/s^2 (blue). Clearly, the larger velocity variance allows the filter to adapt its velocity estimate much quicker in a response to this step function.

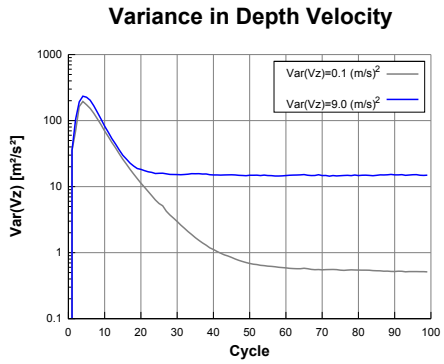

Figure 3.8: Variance of the depth velocity of the differently parametrised Extended Kalman Filters of Figure 3.7.

On the other hand, its variance of the estimation result is much larger, as shown in Figure 3.8. This is also the case, if the point was actually moving at a constant velocity, in which case the model would be correct. The value of the velocity variance has a strong influence on the speed of adaption as well as the quality of the estimate. Bearing in mind that the outlier test normalises the innovation vector by the innovation covariance matrix, a high system variance causes the filter to accept larger innovations than a filter with a small system variance. If the worst comes to the worst, the outlier test is effectively disabled, and the quality of the estimation will be poor. On the other hand, a filter with a small system variance could easily interpret the sudden violation of the model as a series of outliers, and the filter would be reset. So the best value for the system variance is always a trade-off between the speed of adaption and the quality of the estimation, and highly depends on the application and the requirements of the following processing steps.

3.3 Multiple Filters for Improved Speed of Convergence

3.3.1 The Initialisation Problem

When a filter is set up to estimate the motion of an observed world point, the question is how to initialise its unknown state components, in this case its velocity components. Obviously, they can not be derived from a single measurement, or, as we have seen above, from its first two measurements due to the immanent measurement noise.

Usually, the initial state variance of the unknown components is set to a very high value, forcing the filter to ignore the initial values completely. Although this leads to a high speed of convergence on average, this also results in large state estimate variances, since the first measurements have a large impact on the estimated velocity. As the following processing steps require a reasonable velocity estimate, this introduces a large delay until the state estimate variance is sufficient small, contradicting the real-time constraint of the application.

On the other hand, setting the initial state variance of a poorly initialised filter to a small value, the estimated state will converge only slowly to the correct value. So the question is how to determine a good initialisation for a filter, that leads to a correct state estimate and a small state estimate variance. We will refer to this as the *initialisation problem*.

Assuming the filter to initialise belongs to an object whose motion is already known by other converged filters, the new filter can be simply initialised to the objects motion. Since the 6D-Vision algorithm estimates a sparse motion field for individual points without any object knowledge, the question is how to decide which filter belongs to the same object. Without an image based object segmentation, only the spatial vicinity derived from the stereo information can be used to establish this correspondence. However, as we have seen in the introduction, there are cases in which nearby points have quite different motions, and such an approach would easily lead to a wrong initialisation.

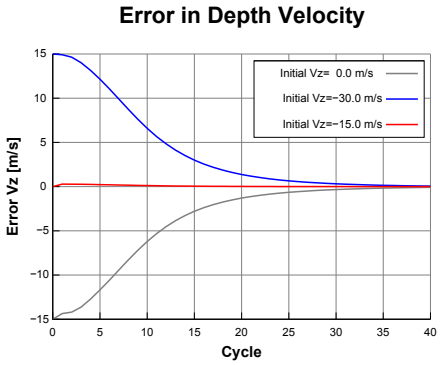

Figure 3.9: Error in depth velocity for the same Extended Kalman Filter initialised with different velocities.

The evolution of differently initialised filters with the same initial state variance is shown in Figure 3.9. Clearly, an initialisation with the correct velocity (red) provides a much better estimate for the first frames, than the wrongly initialised filters (grey and blue). It can also be seen, that, in the beginning, the change of the state of the wrongly initialised filters is much larger than the one of the correctly initialised filter.

To solve the initialisation problem, the idea presented here is to run multiple filters in parallel, each of them initialised or parametrised differently. By using a unified decision strategy, the best matching filter is then picked and given to the following processing steps. This is similar to the particle filter, that uses multiple state hypotheses and picks the one with the highest matching score. In contrast to the particle filter, however, the presented method performs no re-sampling around the found optimum.

3.3.2 A Unified Decision Strategy

Running multiple filters in parallel, the question is how to decide which filter performs best. Since the variance of the state estimate gives only the variance around the state, and not around the correct value, it can not be used as a decision strategy.

A wrongly initialised Kalman Filter has to change its state vector constantly, due to its large innovation vector, which is the difference between the observed measurement and the predicted measurement. From the definition of the Kalman Filter follows, that the probability density function of the innovation vector $\mathbf{s}$ is given by the multivariate normal distribution

$$f(\mathbf{s}) = \frac{1}{(2\,\pi)^{\frac{n}{2}}\,|\mathbf{S}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\mathbf{s}^{\top}\mathbf{S}^{-1}\mathbf{s}\right) \tag{3.32}$$

with $\mathbf{S}$ as the innovation covariance, $|\mathbf{S}|$ as its determinant and n as the number of dimensions. Assuming independent measurements, the likelihood function of a series of innovation vectors $\mathbf{s}_i$ is then given by the product

$$L(\mathbf{s}) = \prod_{i=0}^{k} f(\mathbf{s}_i)\,. \tag{3.33}$$

By taking the logarithm of the likelihood function, this can also be written as the sum

$$\ln L(\mathbf{s}) = \sum_{i=0}^{k}\left(\frac{1}{(2\,\pi)^{\frac{n}{2}}\,|\mathbf{S}_i|^{\frac{1}{2}}}\right) - \frac{1}{2}\sum_{i=0}^{k}\left(\mathbf{s}_i^{\top}\mathbf{S}_i^{-1}\mathbf{s}_i\right). \tag{3.34}$$

If the predicted measurement and the actual one closely match, the innovation vector will be small, and the likelihood function will reach a higher value than for a constantly larger innovation vector. The best matching filter is therefore found as the one with the highest likelihood. Since the likelihood analysis incorporates the innovation covariance matrix, this decision strategy does not only allow the usage of multiple filters with different initialisations, but also filters with different parametrizations.

In Figure 3.10 the values of the probability density function of the three differently initialised Extended Kalman Filters of Figure 3.9 are shown. As

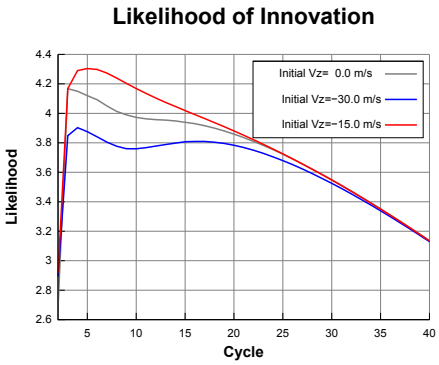

Figure 3.10: The values of the probability density function of the three differently initialised Extended Kalman Filters of Figure 3.9.

it can be seen, the filter with the correct initialisation of $-15.0\,\mathrm{m/s}$ has the highest likelihood during the convergence phase, and would be therefore selected by the multiple filter system.

Instead of summing up the elements from the beginning, one can also calculate the sum over the last m cycles, allowing the filter system to react more quickly in the event of a changed state. To eliminate the need to store the previous likelihoods, it is more convenient to use a low-pass filter, acting as a fading memory. After the initialisation phase, the Kalman Filter system can be reduced to a single Kalman Filter to save computational resources.

3.4 Conclusion

The presented 6D-Vision principle of fusing the optical flow and stereo information in a single Kalman Filter to estimate the three-dimensional motion of a single observed point clearly outperforms differential methods. Due to the recursive estimation strategy of the Kalman Filter, new measurements are added each frame to improve the motion field estimation, without the need of large buffers as required by batch processing methods. Also, the computational costs are minimal, since only one measurement has to be processed by the filter each frame.

As shown, the effect of the linearization error of the non-linear measurement system can be reduced by using the Iterated Extended Kalman Filter. And by using multiple filters running in parallel, the speed of convergence is greatly improved in the initialisation phase.

Running one or more Kalman Filters for each point to analyse may seem to be computational expensive. However, efficient modifications like the Bierman algorithm [73] reduce the computational complexity drastically by avoiding the matrix inversion when calculating the Kalman gain. Using

a code generator that performs various additional optimisations like loop unrolling or neglecting unused matrix elements, an optimal implementation was realised that allows to run 640×480 6D-Vision Kalman Filters in only 12 ms on a GPU[2].

Of course, the 6D-Vision principle estimates only the linear motion field, neglecting higher motion terms and leads therefore to slightly wrong estimations in the case of accelerations. This is acceptable, since the principle was introduced to improve the following object segmentation step, by providing a much richer information source than the stereo or optical flow field alone. Once an object is detected and segmented in the image, higher order terms can be estimated much better by integrating the information over the whole object (see Section 6.2).

[2]NVidia GTX 285.

Chapter 4

Ego-Motion Estimation

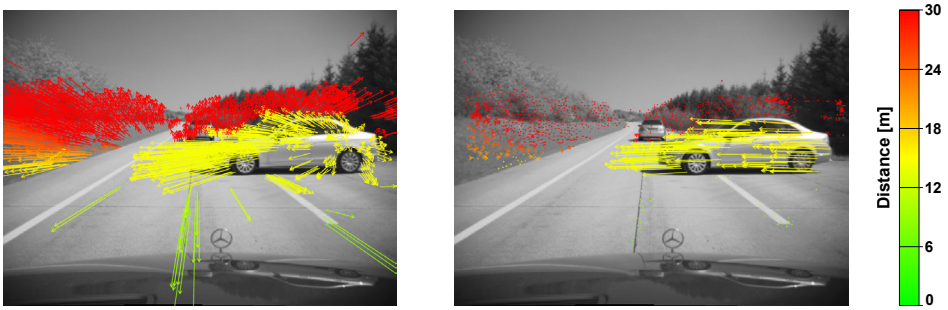

Figure 4.1: Observed 3D motion field (left) and absolute motion field (right).

The presented 6D-Vision principle allows the real-time reconstruction of the 3D motion field observed by a stereo camera system. If the observer moves, like for example a moving car equipped with a stereo camera system, the observed motion field contains not only the motion of the individual objects, but also the motion field induced by the observers own motion. This can be seen in Figure 4.1: The left image shows the observed 3D motion field, whereas the right image shows the motion field of the same scene relative to a fixed world coordinate system. The absolute motion field is not only more intuitive to interpret but also optimally suited for the following processing steps. In order to retrieve such an absolute motion field, the observers motion, also called the *ego-motion*, must be known. It gives the motion of the camera between two successive frames, and is the inverse of the observed scene motion.

In this chapter, an algorithm is presented to estimate the ego-motion based on the observed image sequence. This algorithm takes advantage of the previously reconstructed 3D motion field and works on the same input data as the 6D-Vision algorithm, that is the optical flow field and the stereo information. Since the estimation is performed using a Kalman

Filter, a vehicle model can easily be added to this estimation algorithm. In addition, the data of available inertial sensors is directly integrated into the measurement model.

4.1 Introduction

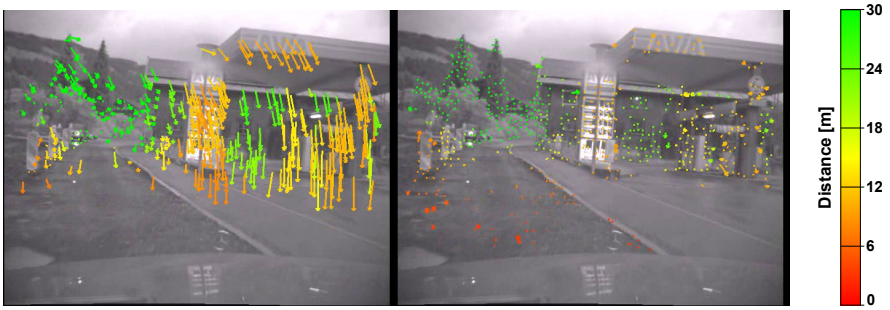

Figure 4.2: Estimation result of the 6D-Vision algorithm with the ego-motion reconstructed from the inertial sensors (left) and the proposed image-based ego-motion estimation algorithm.

To determine the ego-motion of a car, inertial sensors that measure the linear and rotational accelerations of the reference system can be used. However, the quality of the ego-motion reconstruction highly depends on the quality of these sensors, and for a complete motion reconstruction sensors for all degrees of freedom must be available. Modern cars are usually equipped with sensors for the speed and the yaw rate of the vehicle only, that allows the reconstruction of the 2d motion relative to the street, but not the full 3D motion. As a result, the uncompensated ego-motion components induce a virtual movement, as shown in the left image of Figure 4.2, where the car undergoes a significant roll motion. Obviously, such a partial ego-motion reconstruction is not sufficient to obtain the absolute motion field.

Instead of relying solely on the measurements of the inertial sensors, the proposed algorithm to estimate the ego-motion analyses the captured stereo images. In general, image-based ego-motion estimation algorithms work on the assumption that static scene points remain static, and their observed image motion is induced by the motion of the camera. In the literature, many different image-based ego-motion estimation techniques are found. For example, Klappstein et al. presented in [11] a method to estimate the ego-motion from a monocular image sequence. Using only one camera, the translational component of the ego-motion can be determined only up to a scale factor, and therefore at least one inertial sensor is required. Badino presented in [105] an algorithm that matches the point clouds of two successive stereo image pairs using a modified iterative closest point algorithm.

By applying a motion smoothness constraint, Badino was able to increase the accuracy and the robustness of the method even further. For an extensive overview of the existing ego-motion estimation techniques, the reader is referred to [106].

In the context of the proposed system, an ego-motion estimation algorithm has to fulfil the following expectations:

- The algorithm has to be fast, since the complete system has to run at real-time (25 fps). It should therefore use the same input data as the 6D-Vision algorithm and take advantage of the already reconstructed motion field.

- The algorithm should optionally incorporate available inertial sensor data, such as the speed and the yaw rate sensors.

- The movement of a vehicle is constrained by the laws of physic, that can be modelled.

To realise such an algorithm, the proposed method uses an Extended Kalman Filter to estimate the ego-motion [15]. First, static points are identified as points with near zero velocity components from the previously reconstructed motion field. The optical flow and the disparity change of these static points is then used in the measurement model, together with the available inertial sensor data, to update the state vector consisting of the translational and rotational velocities.

4.2 Kalman-Filter based Ego-Motion Estimation

4.2.1 The System Model

The rotational parameters of the ego-motion are the pitch angle α, the yaw angle ψ and the roll angle γ. Their derivative with respect to time are the pitch rate $\dot{\alpha}$, the yaw rate $\dot{\psi}$ and the roll rate $\dot{\gamma}$. The translational parameters are the components of the observers velocity vector $\mathbf{v} = (v_x, v_y, v_z)^\top$. Together with the acceleration a in z-direction and a scale factor β, necessary to compensate systematic errors of the internal speed sensor, the state vector of the system is $\mathbf{x} = \left(\dot{\alpha}, \dot{\psi}, \dot{\gamma}, v_x, v_y, v_z, a, \beta\right)^\top$.

Assuming the rotational and translational velocities remain constant, the

time-continuous system model is given by the equation system

$$\ddot{\alpha} = 0 \tag{4.1}$$
$$\ddot{\psi} = 0 \tag{4.2}$$
$$\ddot{\gamma} = 0 \tag{4.3}$$
$$\dot{v_x} = 0 \tag{4.4}$$
$$\dot{v_y} = 0 \tag{4.5}$$
$$\dot{v_z} = a \tag{4.6}$$
$$\dot{a} = 0 \tag{4.7}$$
$$\dot{\beta} = 0 \tag{4.8}$$

and the time-discrete system model to transform the state vector $\mathbf{x}_{k-1}$ of the previous time step into the current time step k is given by

$$\mathbf{x}_k = \mathbf{A}_k \, \mathbf{x}_{k-1} + \boldsymbol{\omega}_k \tag{4.9}$$

with the state transition matrix defined as

$$\mathbf{A}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

and the noise term $\boldsymbol{\omega}_k$ which is assumed to be Gaussian white noise with the known covariance matrix $\mathbf{Q}_k$.

4.2.2 The Measurement Model

A static world point $\tilde{\mathbf{p}}_k = (x, y, z, 1)_k^\top$ given in homogeneous coordinates in the camera coordinate system at the current time step k is transformed into the previous camera coordinate system at time step $k-1$ by the ego-motion $\mathbf{M}$ according to

$$\tilde{\mathbf{p}}_{k-1} = \mathbf{M}\left(\dot{\alpha}, \dot{\psi}, \dot{\gamma}, v_x, v_y, v_z, \Delta t\right) \, \tilde{\mathbf{p}}_k \tag{4.11}$$
$$= \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}_3^\top & 1 \end{bmatrix} \, \tilde{\mathbf{p}}_k \tag{4.12}$$

with $\mathbf{R}$ as the 3×3 rotation matrix constructed from the Euler angles $\dot{\alpha}\Delta t$, $\dot{\psi}\Delta t$ and $\dot{\gamma}\Delta t$, and $\mathbf{t}$ as the translation vector $\mathbf{t} = \Delta t \; (v_x, v_y, v_z)^\top$. Assuming

the rotation angles are small, the rotation matrix can be approximated as

$$\mathbf{R}\left(\dot{\alpha}\Delta t,\, \dot{\psi}\Delta t,\, \dot{\gamma}\Delta t\right) = \begin{bmatrix} 1 & -\dot{\gamma}\Delta t & \dot{\psi}\Delta t \\ \dot{\gamma}\Delta t & 1 & -\dot{\alpha}\Delta t \\ -\dot{\psi}\Delta t & \dot{\alpha}\Delta t & 1 \end{bmatrix}. \tag{4.13}$$

From the projection formula follows for the observed optical flow and the disparity change:

$$\Delta u = u_k - u_{k-1} = u_k - \left(f_u \frac{x_{k-1}}{z_{k-1}} + u_0\right) \tag{4.14}$$

$$\Delta v = v_k - v_{k-1} = v_k - \left(f_v \frac{y_{k-1}}{z_{k-1}} + v_0\right) \tag{4.15}$$

$$\Delta d = d_k - d_{k-1} = d_k - \frac{b\, f_u}{z_{k-1}}. \tag{4.16}$$

To determine the components of the world point $\tilde{\mathbf{p}}_{k-1}$, the world point $\tilde{\mathbf{p}}_k$ is first calculated from the image measurements $(u, v, d)_k^\top$ by the inverse projection formula as

$$\tilde{\mathbf{p}}_k = \acute{\mathbf{\Pi}}^{-1} \begin{pmatrix} u \\ v \\ d \end{pmatrix}_k \tag{4.17}$$

and then transformed by equation (4.11) into the previous time step, followed by the conversion into the euclidean space. The measurement model is then given by the non-linear relation

$$\mathbf{z}_k = \begin{pmatrix} \Delta u \\ \Delta v \\ \Delta d \end{pmatrix} = h\left(\mathbf{x}, \mathbf{p}_k\right). \tag{4.18}$$

The measurement matrix $\mathbf{H}$ is then the Jacobian of the measurement equations with respect to the state vector elements, with the predicted state as the linearization point. Again, this linearization point can be refined iteratively, e.g., by using the Iterated Extended Kalman Filter. Please note, that this measurement model does not model the uncertainty of the inverse projection to obtain $\mathbf{p}_k$ explicitly. This has to be accounted for when specifying the uncertainties of the optical flow and the disparity change in the measurement covariance matrix of the Kalman Filter.

In addition to the image measurements, data obtained by the inertial sensors can be directly incorporated. Here, we assume the velocity data of a speed sensor and a yaw rate sensor giving the rotation around the cars height axis are available. As a first approximation, the cars motion on a planar street can be modelled as a circular driving under static conditions, with the origin of the car coordinate system on the middle of the rear axis. The velocity is usually obtained by measuring the rotational speed of the

wheels, multiplied with the circumference of the wheel, and assumed to be along the longitudinal direction of the vehicle at all time. Let us denote the measured speed as v_s and the measured yaw rate as $\dot{\psi}_s$. After a given time interval Δt, the car has travelled the distance $v_s\,\Delta t$ and is rotated by $\dot{\psi}_s\,\Delta t$. The radius r of the driven circle is then given by

$$v_s\,\Delta t = r\,\dot{\psi}_s\,\Delta t \tag{4.19}$$

$$r = \frac{v_s}{\dot{\psi}_s} \tag{4.20}$$

and the translation vector of the car

$$\mathbf{t}_c = \frac{v_s}{\dot{\psi}_s} \begin{pmatrix} 1 - \cos\left(\dot{\psi}_s\,\Delta t\right) \\ \sin\left(\dot{\psi}_s\,\Delta t\right) \end{pmatrix} \tag{4.21}$$

gives the displacement in the previous car coordinate system. The effective velocity vector is then given as $\mathbf{v}_e = \frac{1}{\Delta t}\mathbf{t}_c$.

In general, the camera coordinate system does not correspond to the car coordinate system, and the rotational velocity and effective velocity components can not be directly used as measurements of the ego-motion. Using the transformation matrix $\mathbf{M}_1$ to transform a point of the camera coordinate system into the car coordinate system, the ego-motion matrix is obtained from the inertial motion matrix $\mathbf{M}_i$, which is composed of the rotation matrix $\mathbf{R}_c\left(\dot{\psi}_s\,\Delta t\right)$ and the translation vector $\mathbf{t}_c$, as

$$\mathbf{M} = \mathbf{M}_1^{-1}\mathbf{M}_i\mathbf{M}_1. \tag{4.22}$$

Decomposing this motion matrix into the rotational and translational components, the effective rotational and translational velocities in the camera coordinate system are obtained and used as direct measurements.

If the orientation of the two coordinate systems is identical, the rotational velocity can be directly used, and only the translational velocity vector has to be transformed. As a first approximation, the velocity along the z-axis can be directly used, whereas the velocity component along the x-axis suffers from the side slip angle [107] and is given by

$$v_x = \left(c_1 - c_2 v_s^2\right)\dot{\psi}_s \tag{4.23}$$

with the car dependent constants c_1 and c_2.

Finally, the equation

$$0 = v_z - \beta s_z \tag{4.24}$$

describes the relation of the measured velocity along the z-axis of the camera coordinate system s_z and the estimated velocity using the scale factor β.

4.2.3 Selection of the Image Measurements

The model of the described ego-motion estimation assumes that all image measurements correspond to static points. As a first approach, such points are easily determined from the reconstructed motion field of the 6D-Vision algorithm by thresholding the estimated velocities. However, at the time the ego-motion has to be estimated, only the motion field of the previous time step is available. But static points of the previous motion field are not necessarily static in the current motion field. Take for example a standing truck in front of the ego-vehicle. Points on its back side are identified to be static and would therefore be used in the ego-motion estimation. This happens even when the truck starts accelerating, which then causes systematic errors in the estimated ego-motion. To cope with this problem, we will interpret moving points as outliers and concentrate on the outlier detection.

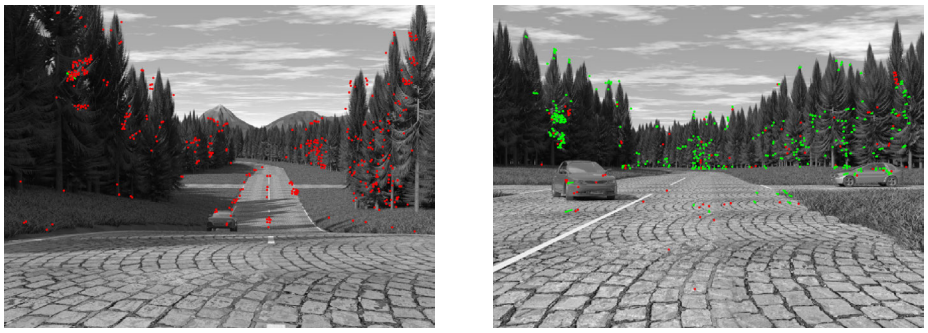

Figure 4.3: Problems of the classic $3\,\sigma$-test for the outlier detection: In the left image, the sudden rotation causes large innovations misinterpreted as outliers (red). In the right image, points on the slowly moving car are misinterpreted as inliers (green).

As stated before, the Kalman Filter is sensitive to measurement outliers, and the presented ego-motion algorithm is no exception. Therefore, a suitable outlier detection has to be established before the measurements are fed into the Kalman Filter. The common approach is to perform a $3\,\sigma$-test, as described in Section 3.2.5. However, this approach has two major drawbacks: First, since the innovation gives the difference between the predicted and the actual measurement, the test may easily fail when the model assumptions are violated or the predicted state is poor. If the worst comes to the worst, all image measurements would be rejected. Secondly, the normalised innovation squared of outliers like slowly moving points may be well below the fixed threshold, resulting in a wrong ego-motion estimation. Both problems are illustrated in Figure 4.3.

Instead of using a large fixed threshold to decide whether a measurement is an inlier or an outlier, which leads to misinterpretation of slowly moving

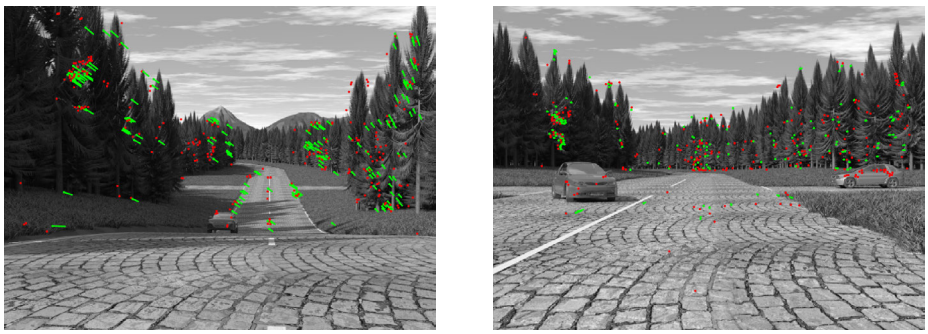

Figure 4.4: Result of the dynamic outlier detection based on the analysis of the normalised innovation squared. Outliers are shown in red, inliers in green.

points, it is more convenient to start with a small threshold, and increase it until a minimum number of measurements is found (typically about 100 points are sufficient). In combination with an Iterated Extended Kalman Filter, this outlier detection step should be repeated after each iteration, since the linearization point and therefore the innovations change each iteration. As it can be seen in Figure 4.4, this method successfully increases the threshold in situations when the model assumptions are violated (left image), but also correctly ignores the points on the slowly moving objects (right image).

In order to speed up the ego-motion estimation, only a subset of the available image measurements should be used. To avoid that the equation system constructed from the image measurements becomes under-determined, the measurements should be equally distributed over the whole image and disparity space. This can be easily implemented using a fixed number of bins and choosing random elements of each bin in turn. In addition, only measurements belonging to static points identified in the previous reconstructed motion field can be used.

4.2.4 Accuracy of the Estimated Ego-Motion

To evaluate the performance of the ego-motion estimation, a synthetic sequence with strong camera motion and multiple moving objects was generated (see Figure 4.5). The KLT-tracker and the correlation stereo were used to obtain a maximum of 1100 image measurements. From these, 400 points were selected equally distributed over the image and the disparity range, without using the 6D-Vision information to obtain static point candidates. This means that the outlier detection has to cope with all the moving points on its own. To evaluate the performance of the image-based estimation, no inertial sensor data was used. The estimation results over the 300 frames of

Figure 4.5: Frame 93 and 104 of the synthetic sequence used to evaluate the performance of the proposed ego-motion estimation.

the sequence are shown in Figure 4.6. Looking at the estimated angular velocities, the strong camera motion becomes obvious. It includes also sudden changes around cycle 130, that clearly violate the constant angular velocity assumption. However, due to the described dynamic outlier detection method, the algorithm is able to cope with these. Overall, the filter clearly manages to estimate the rotational velocities correctly, even in the presence of slowly moving objects that occur at the frames 180-250.

The estimation of the linear velocity components, shown in Figure 4.6, seems to be more noisy than the estimation of the angular velocities. This is not surprising, since the field of view is limited and therefore missing nearby points that would allow a more stable solution. However, looking at the absolute distance error shown in Figure 4.7, the error is mostly well below 1 cm. Overall, the absolute distance errors show no systematic error that would lead to serious estimation errors of the following 6D-Vision analysis.

4.3 Conclusion

The presented ego-motion algorithm is based on the same input data required for the 6D-Vision algorithm, i.e., the optical flow and the stereo information. The processing time of the algorithm is well below 2 ms on a modern customer PC for processing about 400 points, including the binning and outlier detection steps. Due to the dynamic outlier detection, even slowly moving points are identified and not used in the estimation process. Since the underlying Kalman Filter allows modelling even complex vehicle models, it can be easily improved to obtain even better estimation results than the constant velocity model used here. To improve the estimation when large accelerations are expected, the Iterated Extended Kalman Filter should be applied. Depending on the camera motion, 2-5 iterations are sufficient to obtain a stable solution.

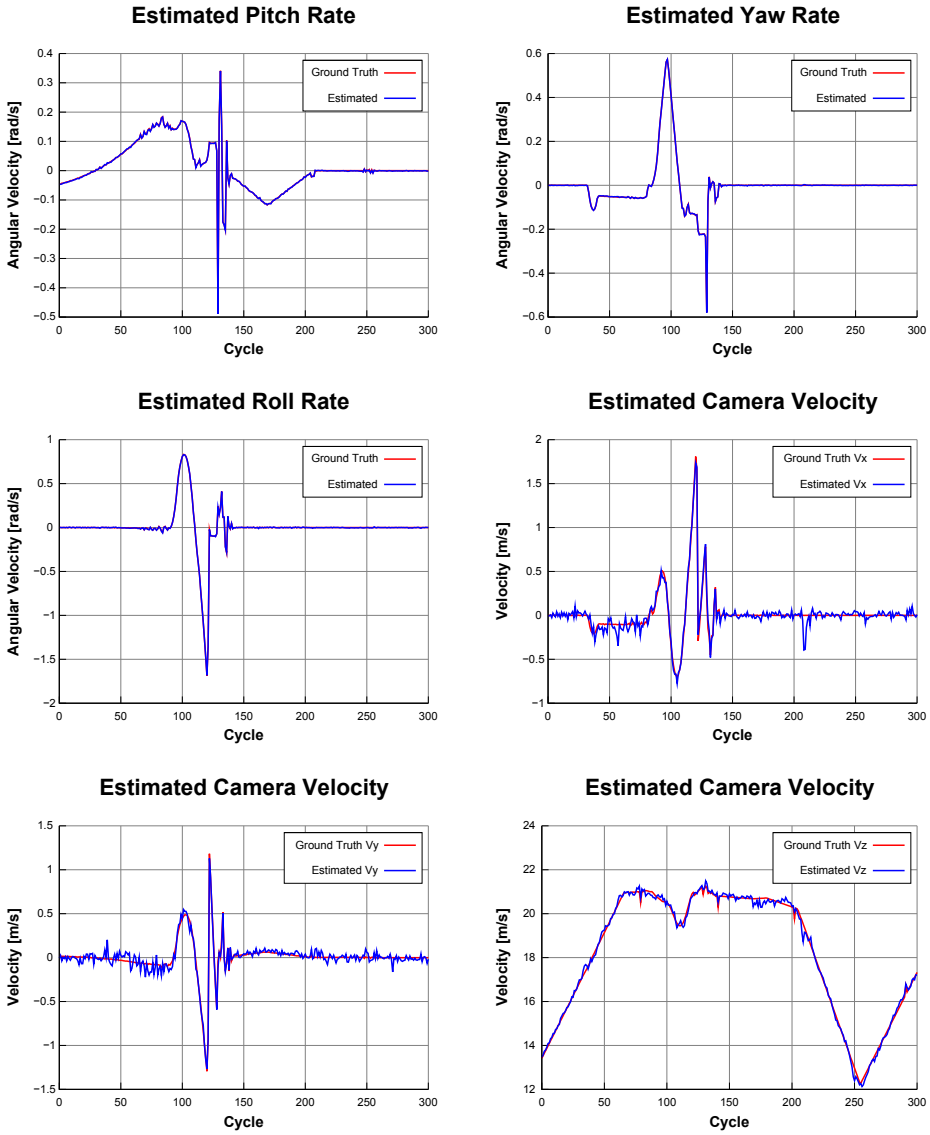

Figure 4.6: Estimated angular and linear velocities of the proposed ego-motion algorithm.

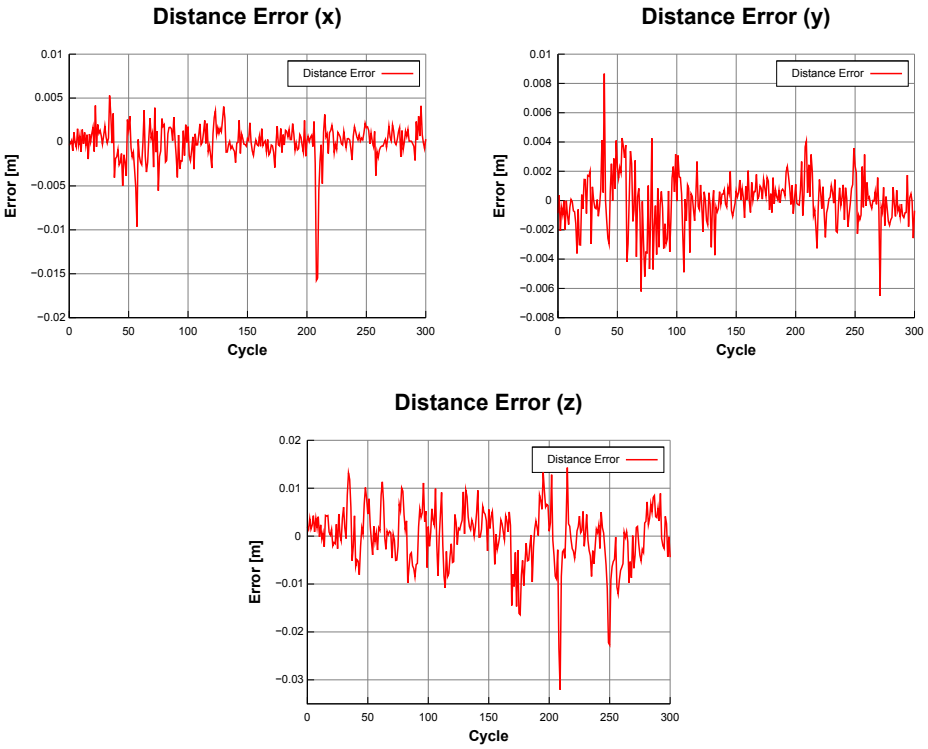

Figure 4.7: Errors of the travelled distance of the ego-motion estimation.

Chapter 5

Variations of the 6D-Vision Principle

Although originally designed to use the KLT tracker to determine the optical flow, and the correlation stereo algorithm to obtain the disparity information, the 6D-Vision principle is not limited on this input information. In this chapter, the application of the 6D-Vision principle to other optical flow and stereo algorithms is described.

5.1 Integration of Flow Fields

Optical flow algorithms provide an optical flow field for each frame, but do not necessarily involve a tracking step. In order to apply the 6D-Vision principle, thus integrating the information over time, correspondences of the individual points have to be established over multiple frames. Here, we will describe the required processing steps to obtain tracks from sparse pixel-discrete and dense sub-pixel-accurate optical flow fields, and compare it to the presented Kanade-Lucas-Tomasi tracker.

5.1.1 Sparse Pixel-discrete Optical Flow Fields

The PowerFlow algorithm of F. Stein [108] deals with the problem of measuring the optical flow in an unconventional manner. Having two images, it first calculates feature descriptors using the census transform [109] for every pixel in each image, and establishes correspondence hypotheses between all pixels whose descriptors match exactly, thus having a Hamming distance of 0. This large number of hypotheses is then filtered according to the discrimination, that is the number of candidates for one particular signature, the similarity of the grey values, and the consistency of the flow field. A big advantage of the PowerFlow algorithm is its robustness against even strong illumination changes, and its ability to cope with large displacement vectors.

A typical resulting flow field is shown in Figure 5.1.

Figure 5.1: Flow field of the PowerFlow algorithm.

Since the PowerFlow algorithm was implemented on a Field Programmable Gate Array (FPGA) integrated in the control unit of the cameras, the flow information is available at real-time without any additional processing costs. Using this flow information in the 6D-Vision algorithm is therefore most desirable. However, there are two problems to deal with: The flow vectors are only pixel discrete, and the sparse flow information has to be transformed into tracks to be able to integrate the information over time.

To establish tracks, a naive approach would simply concatenate the flow vectors, so that the end position of a flow vector of the previous image pair coincides with the start position of a flow vector in the current image pair. Such an approach has not only the drawback that the established tracks would still be pixel-discrete, but suffers also from very short lifetimes of the tracks.

Instead of using only one flow vector, which is pixel-discrete and most likely has no direct successor, it is more convenient to integrate the flow information over a small image region. In the following, such an image region is called feature, in analogy to the KLT. Obviously, the probability of having at least one successive flow vector in such a feature window is much higher, and by using the mean flow vector, sub-pixel accuracy can be easily achieved.

However, calculating the mean flow vector of all flow vectors in the feature window may easily lead to erroneous displacements, not only in the presence of single outliers, but also when the feature is located on flow field boundaries. An alternative would be to use the median flow vector, that is much more robust than the mean, but also much more computational expensive.

The solution presented here combines the benefits of the fast computation of the mean, and the robustness of the median, by splitting the feature into 5 patches, as illustrated in Figure 5.2. First, the mean flow vector of each patch is calculated. Then, the means are compared against each other

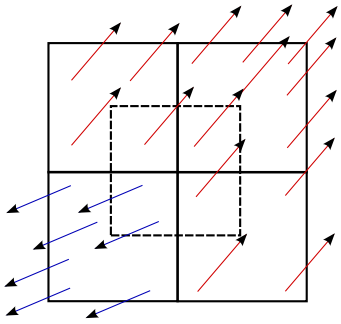

Figure 5.2: Illustration of the spacial flow integration of a single feature using 5 patches.

for compatibility by using a simple thresholding scheme, allowing small variations (usually 1 to 3 pixels) between two compatible patches. Finally, the patches with the highest number of support are used to calculate the resulting mean flow vector, which is used to update the features position. When no consistent flow vector can be found, the feature can not be tracked and is marked accordingly.

Figure 5.3: Resulting feature tracks over the last 3 frames by the spacial integration of the PowerFlow field. The color of the points indicates the age of the track: New points are green and red points indicate a track age of 5 or more frames. The color of the tracks encodes the length of the track in pixels.

Like the KLT, the maximum number of features is fixed and allows to tune the computation time of the tracking step. In order to refill the feature pool when features are lost, or at the start of the tracking process, a feature detection step is performed. With respect to the tracking process, new features should be located at regions of high flow density, where the probability of obtaining flow vectors in the next image is high. With respect to the following processing steps, the features should be equally distributed over the image. To accomplish this, a binary image is created first, marking

the end-points of all flow vectors with 1. Then, a Gaussian filter is applied to this image, giving for every pixel a measure proportional to the number of flow vectors in its neighbourhood. Usually, a 5×5 filter is sufficient. The benefit of a Gaussian filter over a simple mean filter is that the proximity of the flow vectors to a centre pixel is taken into account. By resetting the regions occupied by existing features, including possibly a border, a minimum distance between the existing and new features is assured. Finally, new features are taken at the locations of high flow density indicated by maximas in the image, until the feature pool is refilled or no more candidates remain. The obtained tracks using this method are shown in Figure 5.3.

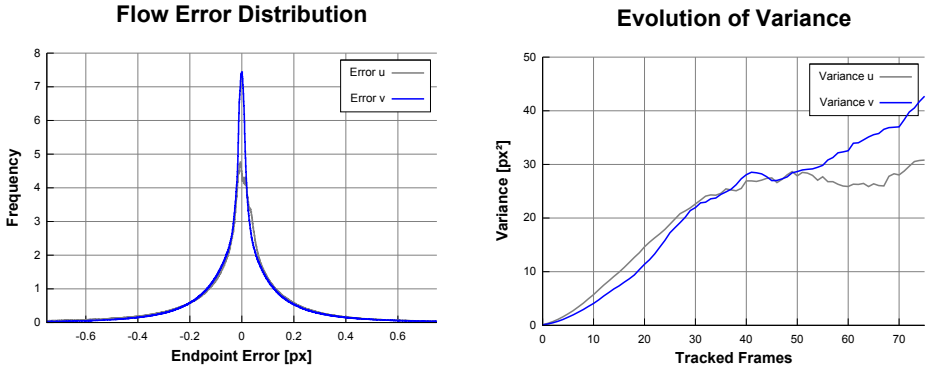

Figure 5.4: Error distribution of the optical flow components u (grey) and v (blue) estimated by the PowerFlow tracker (left), and evolution of the variance over multiple frames (right).

	Non-occluded	Occluded
AEE	0.436 px (σ=0.802 px)	1.162 px (σ=1.383 px)
AAE	6.35° (σ=8.78°)	13.92° (σ=19.59°)
RMS error	0.913 px	1.807 px
R 1°	82.39 %	83.31 %
R 3°	53.77 %	60.09 %
R 5°	38.60 %	49.15 %
R 0.1 px	88.10 %	98.32 %
R 0.5 px	27.45 %	72.36 %
R 1.0 px	6.80 %	41.04 %
Coverage	1.60 %	0.78 %
Computation time	6 ms	

Table 5.1: Evaluation of the PowerFlow tracking algorithm using a maximum number of 10000 features.

Using the same synthetic sequence as in Section 2.3.2, the flow errors

were determined over all frames and evaluated. The overall error measures are shown in Table 5.1, and the error distribution is shown in the left image of Figure 5.4. Compared to the distribution of the KLT tracker (see Figure 2.17 and Table 2.3), the distribution is expanded. The variance of the error in u is $0.059\,\text{px}^2$ ($\sigma_u = 0.25\,\text{px}$) and in v $0.058\,\text{px}^2$ ($\sigma_v = 0.24\,\text{px}$), with their means at about $0\,\text{px}$. Clearly, averaging the vectors yields sub-pixel precision, although it is not as precise as the Kanade Lucas Tomasi tracker. Looking at the right side of Figure 5.4, the evolution of the variance for tracked features increases linear over the first frames, similar to the evolution of the KLT. However, the magnitude of the variance is much larger than for the KLT, due to the higher flow variance.

5.1.2 Dense Optical Flow Fields

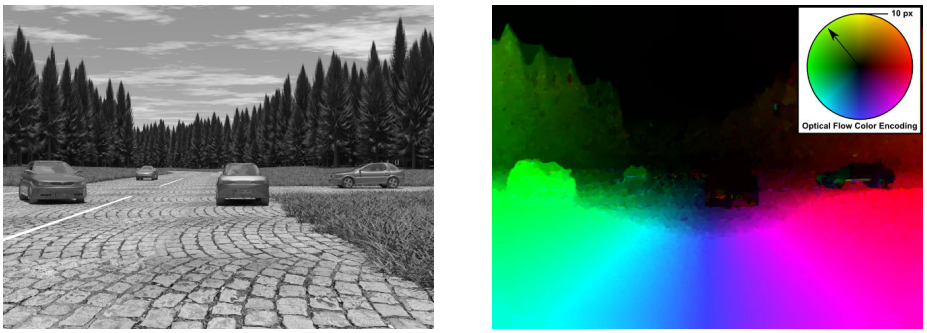

Figure 5.5: Dense optical flow field obtained by the TV-L^1 algorithm. The left image shows the original grey image, the right image encodes the flow vectors according to direction and magnitude.

As pointed out in Section 2.3.2, the gradient constraint derived from the constant brightness assumption yields an under-determined equation system, that cannot be solved without introducing additional constraints. The described Kanade-Lucas-Tomasi tracker solves this problem by assuming locally coherent image motion, resulting in a sparse flow field. Another group of optical flow algorithms provides dense flow information, thus giving an optical flow vector for every pixel in the reference image, by introducing a regularisation term and formulating the problem as an energy minimisation problem. Inspired by the seminal work of Horn and Schunck [60], a diverse range of such techniques has been developed. For a detailed review, the reader is referred to the surveys [110, 111, 112].

The method presented by Horn and Schunck solves the aperture problem by introducing a smoothness constraint, assuming that nearby optical flow vectors are similar in direction and magnitude. Combining the gradient constraint defined by equation (2.30) and the smoothness constraint, the

optical flow field is found by minimising the energy function

$$\begin{aligned} E &= \int_{\Omega} \left(\nabla I\left(\mathbf{x},t\right)^{\top} \mathbf{u}\left(\mathbf{x}\right) + I_1\left(\mathbf{x}\right) - I_0\left(\mathbf{x}\right) \right)^2 \\ &\quad + \alpha^2 \left(\nabla \mathbf{u}\left(\mathbf{x}\right)^{\top} \nabla \mathbf{u}\left(\mathbf{x}\right) \right) d\mathbf{x} \end{aligned} \tag{5.1}$$

with respect to the optical flow vectors $\mathbf{u} = (\delta u, \delta v)^{\top}$. The first term, the gradient constraint, is also called the data term, which measures the dissimilarity of the optical flow solution to the data. The second additive term is called the regularisation or smoothness term, which measures the local dissimilarity of the optical flow. The parameter α defines the weighting between both terms. The minimum of the energy is then found by using the variational method [113], that involves solving the associated Euler-Lagrange equations. Since the gradient constraint is based on a first-order approximation of the image function, the algorithm has to be iterated until it converges or a predefined number of iterations is exceeded.

The obtained optical flow field yields very good results in regions of constant optical flow. However, due to the quadratic penalisation in the smoothness term, the algorithm tends to over-smooth the optical flow field at flow boundaries. Since flow boundaries occur at object boundaries, that usually coincide with large image gradients, Nagel and Enkelmann proposed in [61] to adapt the regularisation according to the local image structure. Although this leads to more accurate flow fields, it does not tackle the root of the problem.

To overcome the over-smoothing problem, Zach et al. presented in [50] an algorithm that solves the energy function

$$\begin{aligned} E &= \int_{\Omega} \lambda \left| I_1\left(\mathbf{x} + \mathbf{u}\left(\mathbf{x}\right)\right) - I_0\left(\mathbf{x}\right) \right| \\ &\quad + \sum_{d=1}^{2} \left| \nabla \mathbf{u}_d\left(\mathbf{x}\right) \right| \, d\mathbf{x} \end{aligned} \tag{5.2}$$

with $\nabla \mathbf{u}_1$, $\nabla \mathbf{u}_2$ as the gradient of the optical flow along the x respifnextchar.. y axis. Since this algorithm is based on the method of total variation and uses the absolute errors rather than the quadratic ones, it is named TV-L^1 algorithm by the authors. Again, using the constant brightness assumption, the data term is linearised and replaced by the gradient constraint, and in order to simplify the minimisation of the energy function, an auxiliary

variable $\mathbf{v}$ is added to the optimisation procedure:

$$E = \int_{\Omega} \lambda \left| \nabla I\left(\mathbf{x},t\right)^{\top} \mathbf{v}\left(\mathbf{x}\right) + I_1\left(\mathbf{x}\right) - I_0\left(\mathbf{x}\right) \right| + \sum_{d=1}^{2} \left[\left|\nabla \mathbf{u}_d\left(\mathbf{x}\right)\right| + \frac{1}{2\Theta}\left(\mathbf{u}_d - \mathbf{v}_d\right)^2 \right] d\mathbf{x} \tag{5.3}$$

with Θ as a small positive constant defining the weight of the coupling term. The introduction of the auxiliary variable allows to split the problem into two steps: First, the data term is solved for a fixed $\mathbf{u}$. Then, the smoothness term is solved for a fixed $\mathbf{v}$. This alternating minimisation procedure is repeated until a maximum number of iterations is reached, or the change in the solution is neglectable [50]. In order to estimate large displacement vectors, an image resolution pyramid is used.

An example of the optical flow field obtained by the algorithm is shown in Figure 5.5. As it can be seen, the flow boundaries are clearly visible. Although the algorithm is computational expensive, it runs in real-time on currently available high-end GPUs [50], using an image pyramid consisting of 5 downscaled images and performing 25 iterations on each downscaled image, and one iteration on the original image.

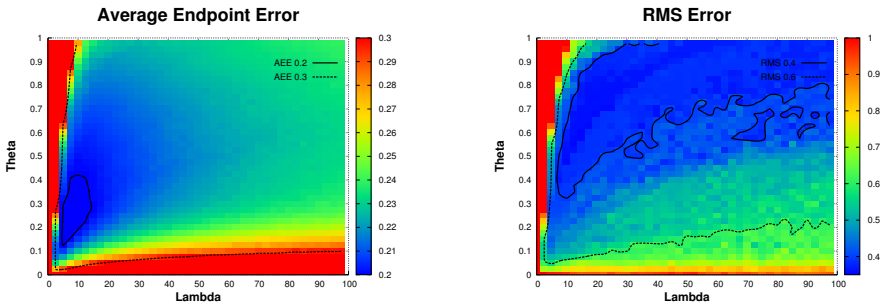

Figure 5.6: Influence of the parameters λ and θ of the TV-L^1 optical flow algorithm on the average endpoint error (left) and the RMS error (right).

The influence of the parameters λ and θ on the average endpoint error and the root mean square error is shown in Figure 5.6. Here, the regions of small average endpoint errors and small RMS errors differ slightly. Obviously, the actual choice of these parameters highly depends on the application. A good compromise for urban scenarios is $\lambda = 10$ and $\theta = 0.35$.

The evaluation of the algorithm on the previously discussed ground truth sequence using these real-time parameters is given in Table 5.2, and the sub-pixel error distribution of the algorithm is shown in Figure 5.7. The variance of the error in u is $0.011\,\mathrm{px}^2$ ($\sigma_u = 0.11\,\mathrm{px}$) and in v $0.008\,\mathrm{px}^2$

	Non-occluded	Occluded
AEE	0.459 px (σ=1.748 px)	1.604 px (σ=1.901 px)
AAE	3.81° (σ=8.71°)	9.06° (σ=17.01°)
RMS error	1.807 px	2.487 px
R 1°	57.12 %	68.36 %
R 3°	28.64 %	43.41 %
R 5°	18.01 %	33.78 %
R 0.1 px	78.20 %	97.52 %
R 0.5 px	14.88 %	70.41 %
R 1.0 px	5.87 %	46.12 %
Coverage	98.53 %	95.17 %
Computation time (GPU)	33 ms	

Table 5.2: Evaluation of the TV-L^1 optical flow algorithm.

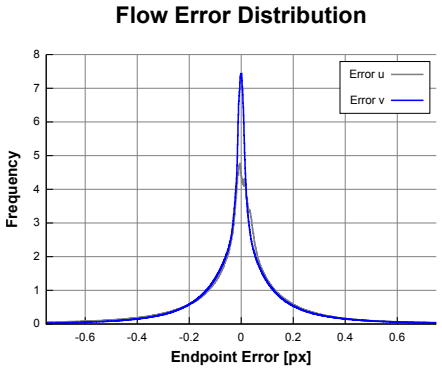

Figure 5.7: Error distribution of the optical flow components u (grey) and v (blue) estimated by the TV-L^1 algorithm.

($\sigma_v = 0.09\,\text{px}$), with their means at about 0 px. Comparing the overall error metrics shown in Table 5.2 with the errors measured by the KLT (see Table 2.3 and Figure 2.17), it seems that the KLT achieves much better results. This is due to the fact, that the KLT calculates the optical flow only at points of rich texture, whereas the dense optical flow algorithm calculates the optical flow for every pixel. Recalling the error variances of the KLT obtained by robust statistics, namely $\sigma_u^2 = 0.016\,\text{px}^2$ ($\sigma_u = 0.13\,\text{px}$) and $\sigma_v^2 = 0.013\,\text{px}^2$ ($\sigma_v = 0.11\,\text{px}$), the benefit of the global method is clearly visible. Obviously, the sub-pixel error variances derived from the robust statistics give a better measure of the accuracy, especially when comparing algorithms of different density.

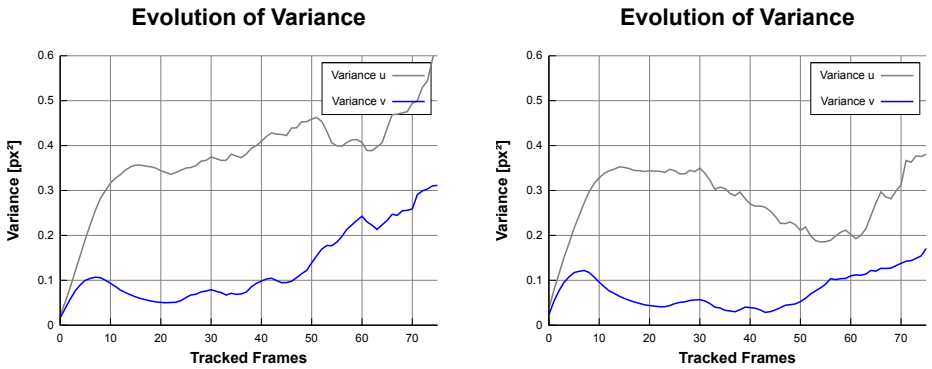

Figure 5.8: Evolution of the variance over multiple frames of the TV-L^1 algorithm using forward (left) and backward calculation (right).

To establish tracks from successive optical flow fields, the flow vectors are simply concatenated and the total displacement is then found by integrating the individual displacements over time. In practice, this involves updating an position image, that contains for each pixel the current sub-pixel position of the associated tracked point. If the optical flow is calculated in the forward direction, which is the optical flow vectors point from the position in the previous image to the position in the current image, we speak of a forward calculation. Here, ambiguities may occur when two or more flow vectors point to the same pixel position in the position image, that have to be resolved. To avoid these ambiguities, the optical flow can be calculated in the backward direction, yielding a flow field with a guaranteed successor.

The evolution of the variance over time is shown in Figure 5.8 for both methods. Comparing it to the results of the KLT, it turns out that the variances of the tracks increase faster for the TV-L^1 algorithm over the first frames, and the variances of the horizontal and vertical flow components differ much more than for the KLT.

5.2 Dense Disparity Maps

Since the 6D-Vision algorithm requires for each feature track a disparity value, a larger number of feature tracks implies also a need for disparity maps of higher density. Dense stereo algorithms provide for nearly every pixel in the input image a disparity value, even in regions of low texture. Like the dense optical flow methods, this is achieved by formulating a global energy function that is then minimised. Here, two real-time capable algorithms are discussed: The Semi-Global Matching and the TV-L^1 stereo algorithm.

5.2.1 Semi-Global Matching

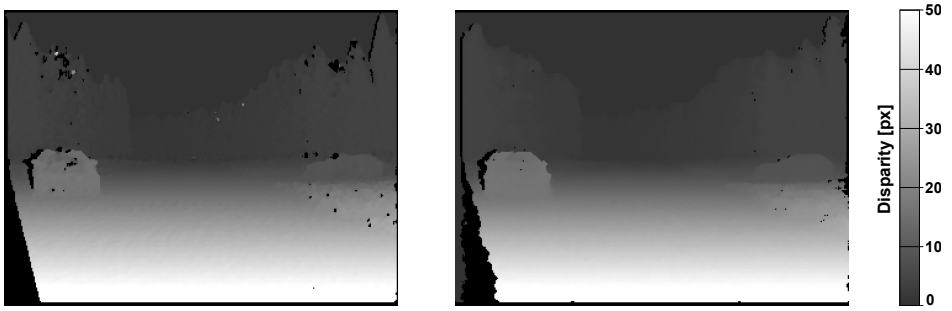

Figure 5.9: Disparity maps of the Semi-Global-Matching algorithms using ZSAD (left) resp. Census matching costs (right).

In [51] Hirschmüller presented an algorithm to obtain dense disparity maps by using mutual information as a cost function and minimising the energy functional

$$\begin{aligned} E &= \sum_{\mathbf{x}\in\Omega} C\left(\mathbf{x}, d_{\mathbf{x}}\right) + \\ &\sum_{\mathbf{x}\in\Omega}\sum_{\mathbf{y}\in N_{\mathbf{x}}} p_1 T\left[\left|d_{\mathbf{y}} - d_{\mathbf{x}}\right| = 1\right] + \\ &\sum_{\mathbf{x}\in\Omega}\sum_{\mathbf{y}\in N_{\mathbf{x}}} p_2 T\left[\left|d_{\mathbf{y}} - d_{\mathbf{x}}\right| > 1\right] \end{aligned} \tag{5.4}$$

with $C\left(\mathbf{x}, d_{\mathbf{x}}\right)$ as the matching cost function of the disparity $d_{\mathbf{x}}$ at the image position $\mathbf{x}$, $N_{\mathbf{x}}$ as the neighbourhood of $\mathbf{x}$, p_1 and p_2 as penalties and $T\left[\blacksquare\right]$ as a function returning 1 if the inner expression is true, and 0 otherwise. The first term simply sums all matching costs, and can be interpreted as the data term. The second and third terms act as a smoothness term: Small deviations of neighbouring disparities are penalised by a penalty p_1, and large deviations are penalised by the penalty p_2. Since the penalty p_1 is

smaller than the penalty p_2, slanted or curved surfaces are preferred over disparity discontinuities.

Since a global solution to the energy minimisation problem is computational expensive, Hirschmüller proposed to solve it approximately using dynamic programming, thus giving it the name Semi-Global-Matching. The recursive scheme for the costs of the applied dynamic programming is defined as

$$\begin{aligned} L_{\mathbf{r}}(\mathbf{x},d) &= C(\mathbf{x},d) + \min \begin{cases} L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},d) \\ L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},d-1)+p_1 \\ L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},d+1)+p_1 \\ L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},i)+p_2 \qquad i<d-1 \\ L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},i)+p_2 \qquad i>d+1 \end{cases} \\ &\quad - \min_k L_{\mathbf{r}}(\mathbf{x}-\mathbf{r},k) \end{aligned} \tag{5.5}$$

along a path defined by the step vector $\mathbf{r}$. The costs over multiple paths of different directions (horizontal, vertical and diagonal) are accumulated and the resulting disparity is then found as the one with the minimum accumulated cost. Since paths of different directions are used, the typical streaking effects known of stereo algorithms evaluating only one path can be removed almost completely.

The resulting disparity map is only pixel-discrete, since the costs are only calculated for discrete disparity values. To obtain sub-pixel accuracy, the costs around the found minimal disparity are taken and the refined disparity is then found at the minimum of the parabola through these points.

Although the original algorithm used mutual information to determine the costs of a disparity match, the energy minimisation scheme can be used with almost any matching score. In practice, the zero-mean sum of absolute differences (ZSAD) and the Census operator [109] proved to be most robust even in situations of large illumination changes, and are on the other hand less computational expensive than the mutual information measure.

Solving the minimisation problem still remains computational expensive, and a classical implementation takes about 2 seconds to compute a VGA disparity map on a modern computer. Since the calculation of the cost cube requires access to the predecessors, each path must be computed sequentially, rendering it unsuitable for massive parallel machines like GPUs. However, Gehrig proposed in [114] an implementation on an FPGA, which is able to calculate the disparity map in about 30 ms. The massive speed-up was achieved by calculating the disparity map on a sub-sampled image of half the resolution (overview image), and combine it with the disparity map calculated for a portion of the image calculated at the full resolution (fine image). The implemented engine allows the computation of 64 disparity steps, which leads to a total disparity range of 128 pixels. The implementation supports the ZSAD and Census matching costs, and the sub-pixel refinement is

performed using an equiangular fit. The sub-pixel accuracy of the engine is 1/16 pixel due to the use of fixed-point arithmetic. To remove mismatches, a right-left verification step is performed additionally. The disparity maps of this hardware implementation is shown in Figure 5.9.

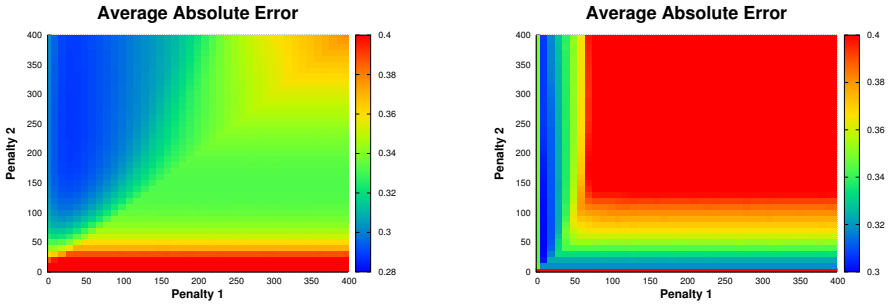

Figure 5.10: Influence of the penalties of the Semi-Global-Matching algorithm on the average absolute error using ZSAD (left) and Census (right) matching costs.

The influence of the penalties on the average absolute error is shown in Figure 5.10. Here, all combinations were tested, although one usually chooses $p_1 < p_2$. As it can be seen, the range of reasonable values for the ZSAD version is much larger than for the Census version. This is not surprising, since the ZSAD measures grey value differences, whereas the Census version uses the Hamming distance between two Census descriptors. Therefore, the range of reasonable values for p_1 of the Census version is limited to a maximum value of about 20. Typical values for the ZSAD version are $p_1 = 10$, $p_2 = 200$ and for the Census version $p_1 = 20$, $p_2 = 100$, when using images with a grey value resolution of 12 bit.

The distribution of the obtained disparities is given in Figure 5.11. As it can be seen, it consists of two distributions, the larger one representing the distribution obtained from the overview image, and the smaller distribution in the range between the disparities 22 and 24.5 coming from the fine image. Due to the limited resolution of the disparity, the actual discretization of disparities of the overview image is 1/8 pixel, whereas the fine image has a resolution of 1/16 pixel. Comparing the Census and the ZSAD version, the pixel locking effect seems much more pronounced on the Census version. Obviously, the assumption of linear increasing costs does not hold for the Census version.

The evaluation of the errors over the ground truth sequence is summarised in the Tables 5.3 and 5.4. Here, the Census version outperforms the ZSAD version in terms of accuracy, but does not reach the accuracy of the correlation stereo method (see Table 2.1). However, one has to keep in mind that the errors were evaluated using the FPGA implementation of the SGM

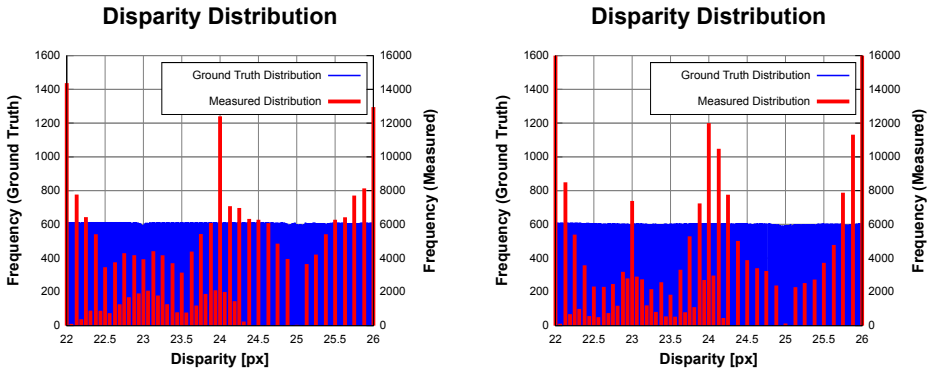

Figure 5.11: Pixel-locking effect of the equiangular sub-pixel interpolation method of the Semi-Global-Matching algorithm using ZSAD (left) and Census (right) matching costs.

	Non-occluded	Occluded
AAE	0.509 px (σ=3.628 px)	6.612 px (σ=18.525 px)
RMS error	3.663 px	19.669 px
R 0.5	15.92 %	76.64 %
R 0.75	8.34 %	69.36 %
R 1.0	5.48 %	63.12 %
R 1.5	3.39 %	52.48 %
R 2.0	2.52 %	43.68 %
Coverage	87.19 %	57.43 %
Computation time	30 ms	

Table 5.3: Evaluation of the Semi-Global-Matching algorithm using the ZSAD cost function.

	Non-occluded	Occluded
AAE	0.432 px (σ=2.111 px)	6.526 px (σ=19.401 px)
RMS error	2.155 px	20.469 px
R 0.5	15.47 %	79.73 %
R 0.75	9.03 %	72.09 %
R 1.0	6.72 %	65.49 %
R 1.5	4.47 %	54.96 %
R 2.0	3.16 %	45.75 %
Coverage	87.56 %	64.40 %
Computation time	30 ms	

Table 5.4: Evaluation of the Semi-Global-Matching algorithm using the Census operator.

algorithm, that was optimised to achieve real-time, whereas the presented results for the correlation algorithm correspond to a CPU implementation optimised with respect to accuracy.

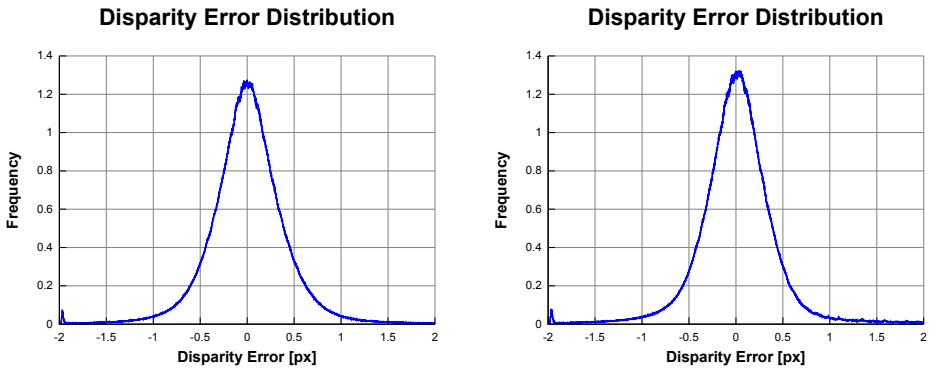

Figure 5.12: Error distribution of the Semi-Global-Matching algorithm using ZSAD (left) and Census (right) matching costs.

The error distributions obtained by accumulating the errors over the 681 frames ground truth sequence are shown in Figure 5.12. The mean of the distribution of the ZSAD version is at 0.004 px, whereas the mean of the Census version lies at 0.010 px. The variances are nearly identical: $\sigma^2 = 0.10\,\mathrm{px}^2$ ($\sigma = 0.32$ px) for the ZSAD and $\sigma^2 = 0.09\,\mathrm{px}^2$ ($\sigma = 0.30$ px) for the Census version.

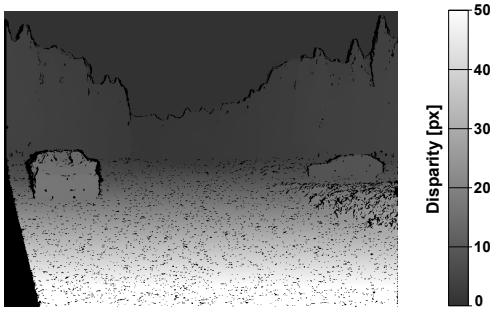

Figure 5.13: Disparity map of the TV-L^1 stereo algorithm.

5.2.2 TV-L^1 Stereo

Inspired by the promising results of the TV-L^1 optical flow algorithm, it is only reasonable to adapt the algorithm to the stereo case and evaluate its performance. Using a standard stereo configuration, the energy functional to minimise with respect to the disparity map $d(\mathbf{x})$ is defined as

$$E = \int_{\Omega} \lambda \left| I_l(\mathbf{x}) - I_r(\mathbf{x} - d(\mathbf{x})) \right| + \sum_{d=1}^{2} \left| \nabla d_d(\mathbf{x}) \right| \, d\mathbf{x}. \tag{5.6}$$

This energy is then minimised using the same approach as described in Section 5.1.2, by introducing the auxiliary variable p and solving the energy

$$\begin{aligned} E &= \int_{\Omega} \lambda \left| I_l(\mathbf{x}) - I_r(\mathbf{x}) + \frac{\partial I}{\partial u}(\mathbf{x}) \, p(\mathbf{x}) \right| \\ &\quad + \sum_{d=1}^{2} \left[\left| \nabla d_d(\mathbf{x}) \right| + \frac{1}{2\Theta} \left(d_d - p_d \right)^2 \right] d\mathbf{x} \end{aligned} \tag{5.7}$$

alternately for the data and smoothness term.

The algorithm yields a disparity value for every pixel in the left image, including occluded areas where an actual measurement is impossible. To improve the quality of the algorithm and eliminate most of the occluded pixels, a right-left verification step is performed: The disparity map is warped from the left to the right image and used as an initialisation to perform a few additional iterations in the reverse order. The disparity maps of the left and the right image are then compared, and inconsistencies are marked accordingly.

This algorithm was implemented on the GPU by the author, based on the already available implementation of the TV-L^1 optical flow algorithm. The resulting disparity map for the ground truth image is shown in Figure 5.13.

The influence of the parameters λ and θ on the average absolute error and the root mean square error is shown in Figure 5.14. Here, the minima

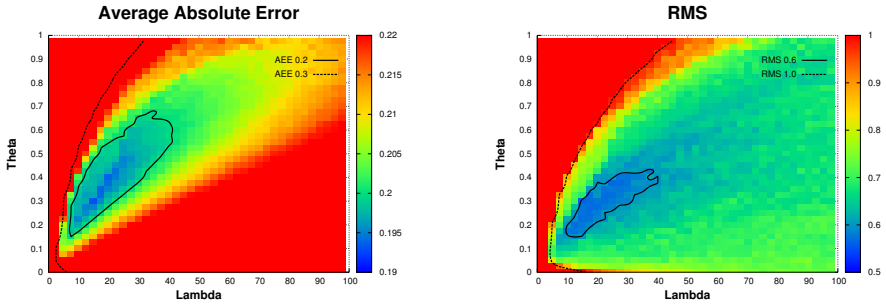

Figure 5.14: Influence of the parameters λ and θ of the TV-L^1 stereo algorithm on the average absolute error (left) and the RMS error (right).

are much more pronounced than for the optical flow (see Figure 5.6). For the following evaluation, an image pyramid consisting of 7 levels in total was used performing 25 iterations on each level, λ was set to 15 and θ to 0.3.

	Non-occluded	Occluded
AAE	0.222 px (σ=1.151 px)	3.95 px (σ=9.23 px)
RMS error	1.172 px	10.037 px
R 0.5	6.29 %	72.24 %
R 0.75	4.24 %	66.46 %
R 1.0	3.29 %	60.82 %
R 1.5	2.42 %	49.64 %
R 2.0	1.95 %	41.92 %
Coverage	94.73 %	49.34 %
Computation time	32 ms	

Table 5.5: Evaluation of the TV-L^1 stereo algorithm.

The evaluation results of the algorithm on the ground truth sequence is shown in Table 5.5. Comparing it to the results of the correlation stereo (see Table 2.1) and the Semi-Global-Matching algorithm (see Table 5.3), this method ranks best in terms of accuracy and coverage. Due to the implementation on the GPU, its computation time is only 32 ms, rendering it well-suited for real-time applications.

In contrast to the other described methods, the TV-L^1 stereo algorithm does not suffer from pixel-locking effects, as it can be seen in the left image of Figure 5.15. The analysis of the disparity error distribution shown in the right image of Figure 5.15 reveals the mean error at -0.01 px and a variance of $\sigma^2 = 0.010\,\mathrm{px}^2$ ($\sigma = 0.099\,\mathrm{px}$), i.e., about a tenth of the variance of the hardware implementation of the Semi-Global-Matching algorithm.

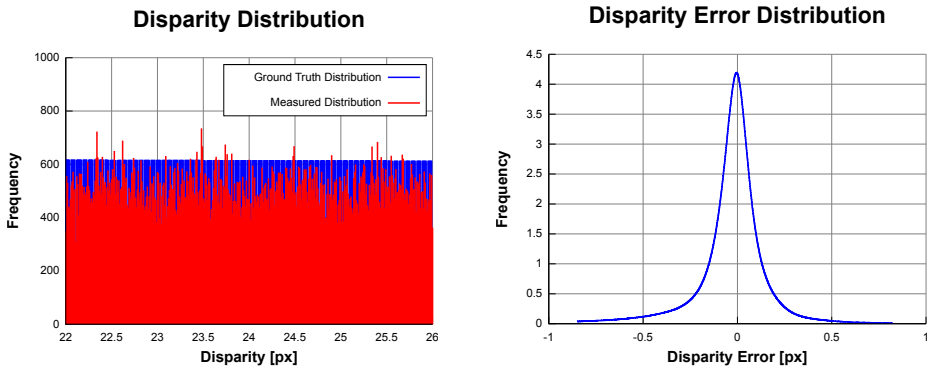

Figure 5.15: Disparity and disparity error distribution of the TV-L^1 stereo algorithm.

5.3 Exploiting the Epipolar Constraint

The standard approach to apply the 6D-Vision principle is to track features in the left or right image, and determine the disparity for each feature by analysing the stereo image pair. However, both image analysis tasks are not perfect, and although the 6D-Vision algorithm copes with the measurement noise well, it is still sensitive to outliers during its initialisation phase. In order to improve the quality of the input data, it is therefore reasonable to incorporate as much information as possible in the image analysis step.

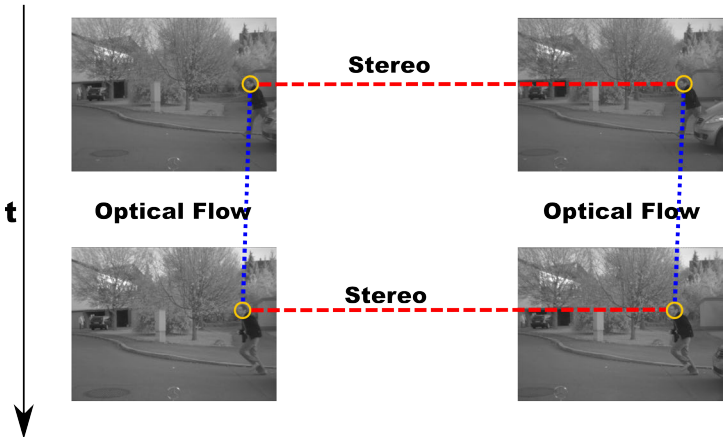

Figure 5.16: Motion and stereo constraints of two stereo image pairs.

Looking at the simplest approach to establish feature tracks including the disparity information illustrated in Figure 5.16, a total of four images has to be analysed: The stereo image pair at the previous time step and the image pair at the current time step. However, each method uses only two

images at a time to obtain the necessary information. In this section, we will therefore discuss how to take advantage of the unused motion constraint and to extend the image analysis task to incorporate all four images at once.

5.3.1 Outlier Rejection

Looking at Figure 5.16, we see two motion and two stereo constraints. The standard approach on the other side uses only one motion constraint, i.e., the optical flow in the left images, and the two stereo constraints independently from each other. The remaining motion constraint, in this case the optical flow in the right images, can easily be calculated using the same optical flow algorithm and used to verify the obtained measurements.

In the standard stereo configuration, the projection of the world point $\mathbf{w}(t-1)$ in the left image at time step $t-1$ is the image point $\mathbf{m}_l(t-1) = (u_{t-1}, v_{t-1})^\top$. Its corresponding projection in the right image is then given by $\mathbf{m}_r(t-1) = (u_{t-1} - d_{t-1}, v_{t-1})^\top$ with the disparity measured by the stereo algorithm. At the next time step, the correspondence between the image points $\mathbf{m}_l(t-1)$ and $\mathbf{m}_l(t) = (u_{t-1} + \Delta u_l, v_{t-1} + \Delta v_l)^\top$ is established by the optical flow algorithm, by measuring the displacement vector $(\Delta u_l, \Delta v_l)^\top$. The stereo algorithm reveals the position of the corresponding right image point $\mathbf{m}_r(t) = (u_{t-1} + \Delta u_l - d_t, v_{t-1} + \Delta v_l)^\top$ by measuring the new disparity d_t. By introducing the disparity change $\Delta d = d_t - d_{t-1}$, the optical flow displacement vector in the right image originating from $\mathbf{m}_r(t-1)$ is therefore given as:

$$\Delta u_r = \Delta u_l - \Delta d \tag{5.8}$$

$$\Delta v_r = \Delta v_l. \tag{5.9}$$

Comparing this expected flow vector with the measured one allows to verify the consistency between all constraints.

	Good Conditions		Bad Conditions	
	No Test	With Test	No Test	With Test
Disparity (d)	5.17 %	4.09 %	8.54 %	5.25 %
Optical Flow (Δu)	2.36 %	1.57 %	7.68 %	2.24 %
Optical Flow (Δv)	1.23 %	0.72 %	5.12 %	1.05 %

Table 5.6: Percentage of outliers without and with the proposed consistency check.

Table 5.6 shows the effect of the consistency check on the outlier percentage, evaluated over the ground truth sequence. Here, a combination of the KLT tracker and the correlation stereo was used. In the first test, standard parameters were used that give good results in typical daylight scenarios.

In the second test, the threshold of the corner detector of the KLT was reduced, to simulate effects occurring in bad weather situations, in which the threshold must be reduced to ensure enough image measurements. Outliers are here identified as a deviation from the ground truth value by more than 1 px. In both tests the percentage of the outliers was reduced for all three evaluated components: d, Δu and Δv. The amount of the reduction is naturally higher in the second test case, since much more outliers were present before the verification test took place. However, this shows that the quality of the image measurements and as a consequence the quality and therefore the robustness of the obtained motion field of the 6D-Vision algorithm can be significantly improved by this consistency check, especially in the case of bad conditions.

5.3.2 Sparse Scene Flow

Using the additional motion constraint for outlier detection allows to apply the already existing methods of optical flow and stereo image analysis. However, to take full advantage of the additional information, here the right image sequence, it is necessary to extend the image analysis to more than only two images. The benefit of such an approach is obvious: Not only does it reveal the optical flow and the stereo information at the same time using only one algorithm, but it also integrates more information and is therefore less sensitive to measurement noise. We refer to these algorithms as scene flow algorithms, since they reveal all the necessary data to reconstruct the scene motion [12]. Here, an extension of the described Kanade-Lucas-Tomasi tracker (see Section 2.3.2) is presented, that performs the feature tracking in the left image sequence, while estimating the disparity for each tracked feature at the same time. Since this algorithm estimates the optical flow and disparity components only for a limited number of features, it is called sparse scene flow algorithm.

From the constant brightness assumption, it follows that the intensity of the projection of a world point is the same in each of the four images. For the optical flow in the left image, the relation

$$I_l\left(\mathbf{x}_{t-1}, t-1\right) = I_l\left(h\left(\mathbf{x}_{t-1}, \mathbf{p}\right), t\right) \tag{5.10}$$

is used, and for the stereo analysis, the relations

$$I_l\left(\mathbf{x}_{t-1}, t-1\right) = I_r\left(g\left(\mathbf{x}_{t-1}, d_{t-1}\right), t-1\right) \tag{5.11}$$

$$I_l\left(\mathbf{x}_t, t\right) = I_r\left(g\left(\mathbf{x}_t, d_t\right), t\right) \tag{5.12}$$

are exploited, with the image motion functions defined as

$$h\left(\mathbf{x}, \mathbf{p}\right) = \mathbf{x} + \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} \tag{5.13}$$

$$g\left(\mathbf{x}, d\right) = \mathbf{x} - \begin{pmatrix} d \\ 0 \end{pmatrix}. \tag{5.14}$$

Let us assume, that the disparity d_{t-1} associated with an image point $\mathbf{x}$ at time step $t-1$ is already known, and the question is how to determine the displacement vector $(\delta u, \delta v)^\top$ and the disparity d_t at the same time. Using equation (5.10), the flow vector can be retrieved directly, as shown in Section 2.3.2. But to apply the stereo constraint of equation (5.12), the new position and therefore the displacement vector must be already known, thus eliminating the possibility of solving the problem in a single step.

A possible solution is to combine the optical flow constraints of the left and the right image, where the relation of the flow in the right image is formulated as

$$\begin{aligned} I_r\left(g\left(\mathbf{x}_{t-1}, d_{t-1}\right), t-1\right) &= I_r\left(g\left(\mathbf{x}_t, d_t\right), t\right) \\ &= I_r\left(g\left(h\left(\mathbf{x}_{t-1}, \mathbf{p}\right), d_t\right), t\right). \end{aligned} \tag{5.15}$$

This formulation was used by Morat et al. in [115] in a variation of the Kanade-Lucas-Tomasi tracker to track large image regions. However, when employed in a recursive manner, the disparity may easily drift since the left and right u components of the individual flows are calculated independently from each other.

A better approach to estimate the optical flow and the disparity simultaneously is to use the relation

$$I_l\left(\mathbf{x}_{t-1}, t-1\right) = I_r\left(g\left(h\left(\mathbf{x}_{t-1}, \mathbf{p}\right), d_t\right), t\right) \tag{5.16}$$

instead. It states that the grey values of the feature in the left previous image are equal to the ones in the current right image. Since the disparity is measured each frame as an absolute value, it does not suffer from the drifting problem when using only the right optical flow. Together with the optical flow relation defined by equation (5.10), the combined error function is then defined as

$$\begin{aligned} E = &\sum_W \left[I_l\left(h\left(\mathbf{x}_{t-1}, \mathbf{p}\right), t\right) - I_l\left(\mathbf{x}_{t-1}, t-1\right)\right]^2 + \\ &\sum_W \left[I_r\left(g\left(h\left(\mathbf{x}_{t-1}, \mathbf{p}\right), d_t\right), t\right) - I_l\left(\mathbf{x}_{t-1}, t-1\right)\right]^2 \end{aligned} \tag{5.17}$$

over the feature window W. The minimum of the error function is found by setting its first derivative to 0. Inserting the image motion equations (5.13) and (5.14), substituting $u_l - d$ by u_r, and performing a first order Taylor approximation of the image functions, we obtain the linear equation system

$$\begin{bmatrix} \frac{\partial I_l}{\partial u} & 0 & \frac{\partial I_l}{\partial v} \\ 0 & \frac{\partial I_r}{\partial u} & \frac{\partial I_r}{\partial v} \end{bmatrix} \begin{pmatrix} \delta u_l \\ \delta u_r \\ \delta v \end{pmatrix} = \begin{pmatrix} \Delta I_l \\ \Delta I_{lr} \end{pmatrix} \tag{5.18}$$

with

$$\begin{aligned} \Delta I_l &= I_l(u_l, v, t-1) - I_l(u_l, v, t) \\ \Delta I_{lr} &= I_l(u_l, v, t-1) - I_r(u_r, v, t). \end{aligned}$$

This equation system is under-determined for a single image point, since there are only two equations for the three parameters. Therefore, the information is integrated over a small image feature, assuming constant image motion over the feature window. Like with the classical KLT, the parameter vector $(\delta u_l, \delta u_r, \delta v)^\top$ is then found as the solution of

$$\begin{pmatrix} \delta u_l \\ \delta u_r \\ \delta v \end{pmatrix} = \begin{bmatrix} I^l_{uu} & 0 & I^l_{uv} \\ 0 & I^r_{uu} & I^r_{uv} \\ I^l_{uv} & I^r_{uv} & I^l_{vv} + I^r_{vv} \end{bmatrix}^{-1} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \tag{5.19}$$

with the components

$$\begin{aligned} I^l_{uu} &= \sum w(\mathbf{x})^2 \frac{\partial I_l}{\partial u}\frac{\partial I_l}{\partial u} \\ I^l_{uv} &= \sum w(\mathbf{x})^2 \frac{\partial I_l}{\partial u}\frac{\partial I_l}{\partial v} \\ I^l_{vv} &= \sum w(\mathbf{x})^2 \frac{\partial I_l}{\partial v}\frac{\partial I_l}{\partial v} \\ I^r_{uu} &= \sum w(\mathbf{x})^2 \frac{\partial I_r}{\partial u}\frac{\partial I_r}{\partial u} \\ I^r_{uv} &= \sum w(\mathbf{x})^2 \frac{\partial I_r}{\partial u}\frac{\partial I_r}{\partial v} \\ I^r_{vv} &= \sum w(\mathbf{x})^2 \frac{\partial I_r}{\partial v}\frac{\partial I_r}{\partial v} \\ b_1 &= \sum w(\mathbf{x})^2 \frac{\partial I_l}{\partial u}\Delta I_l \\ b_2 &= \sum w(\mathbf{x})^2 \frac{\partial I_r}{\partial u}\Delta I_{lr} \\ b_3 &= \sum w(\mathbf{x})^2 \left(\frac{\partial I_l}{\partial v}\Delta I_l + \frac{\partial I_r}{\partial v}\Delta I_{lr}\right). \end{aligned}$$

Since the application of the Taylor approximation is only verified in the proximity of the solution, the linearization is iterated until the update of the parameter vector is sufficiently small. In order to cope with large displacement vectors, the algorithm is performed on an image pyramid.

As with the original KLT, good features to track are those for which the equation system is well defined, which means that the eigenvalues of the matrix to invert must be large enough. But at the time we are looking for new features, we do not necessarily know the stereo correspondence. So we need an initial guess of the image gradients in the right image. As we assume that the image patch looks exactly the same in the left and the right

image, we can assume that they have also the same gradients. Using this fact, the eigenvalues of the matrix are calculated as

$$\lambda_{1,2} = \frac{I_{uu} + 2I_{vv} \pm \sqrt{(I_{uu} - 2I_{vv})^2 + 8I_{uv}^2}}{2} \tag{5.20}$$

$$\lambda_3 = I_{uu}. \tag{5.21}$$

and new features are initialised at locations where the minimum eigenvalue is above a predefined threshold. To initialise the disparity, respifnextchar.. the position in the right image, a simple stereo algorithm can be used (see Section 2.3.1), or the given algorithm is adjusted to the stereo case by setting the last image pairs to the current ones, and estimating only the position in the right image. This way, the same algorithm can be used to perform the feature tracking and the initialisation, without the need of an extra stereo algorithm.

To eliminate mismatches, a verification check is performed after the optical flow components and the disparity is estimated. Here, the zero mean sum of squared differences between the final feature windows is calculated and thresholded against a predefined value. If it exceeds the threshold, the feature is relocated using the described initialisation technique.

	Non-occluded	Occluded
AEE	0.239 px (σ=0.740 px)	1.017 px (σ=1.402 px)
AAE	2.844° (σ=7.156°)	10.513° (σ=19.125°)
RMS error	0.777 px	1.732 px
R 1°	48.41 %	59.56 %
R 3°	19.97 %	37.08 %
R 5°	11.41 %	30.11 %
R 0.1 px	64.53 %	90.22 %
R 0.5 px	8.08 %	56.86 %
R 1.0 px	3.06 %	34.08 %
Coverage	1.34 %	1.24 %
Computation time (GPU)[1]	35 ms	

Table 5.7: Evaluation of the optical flow components of the proposed sparse scene flow algorithm.

The evaluation of this algorithm on the ground truth sequence is split into the optical flow part shown in Table 5.7 and the stereo part in Tables 5.8

[1]The tracker was evaluated on a NVidia GTX 285 graphics card with a maximum number of 10000 features. The computation time includes the preprocessing and tracking steps, but excludes the feature detection step.

	Non-occluded	Occluded
AAE	0.25 px (σ=0.57 px)	1.56 px (σ=1.62 px)
RMS error	0.619 px	2.247 px
R 0.5	9.50 %	65.31 %
R 0.75	6.12 %	60.25 %
R 1.0	4.79 %	54.99 %
R 1.5	3.33 %	42.21 %
R 2.0	2.24 %	31.79 %
Coverage	1.09 %	0.62 %

Table 5.8: Evaluation of the initial disparity error of the proposed sparse scene flow algorithm.

	Non-occluded	Occluded
AAE	0.26 px (σ=0.68 px)	1.57 px (σ=1.64 px)
RMS error	0.726 px	2.269 px
R 0.5	9.99 %	65.80 %
R 0.75	6.38 %	60.50 %
R 1.0	4.96 %	55.11 %
R 1.5	3.46 %	42.25 %
R 2.0	2.32 %	31.65 %
Coverage	1.06 %	0.61 %

Table 5.9: Evaluation of the disparity error after one frame of the proposed sparse scene flow algorithm.

and 5.9. Here, 10000 features were used. The optical flow results are comparable with the ones of the KLT (see Table 2.3). This is not surprising, since the method can be seen as an extension of the classical Kanade-Lucas-Tomasi algorithm to the stereo case. The initial disparity error shown in Table 5.8 gives the error distribution obtained when initialising the disparity by the same algorithm that is used for the tracking. Here, we get much better results than the correlation stereo algorithm (see Table 2.1), and slightly worse results than the dense TV-L^1 stereo (see Table 5.5). As it can be seen in Table 5.9, the disparity error after one frame is quite similar to the error of the initialisation. This was expected, since the disparity is calculated each frame as an absolute measure, and does not suffer from integration effects.

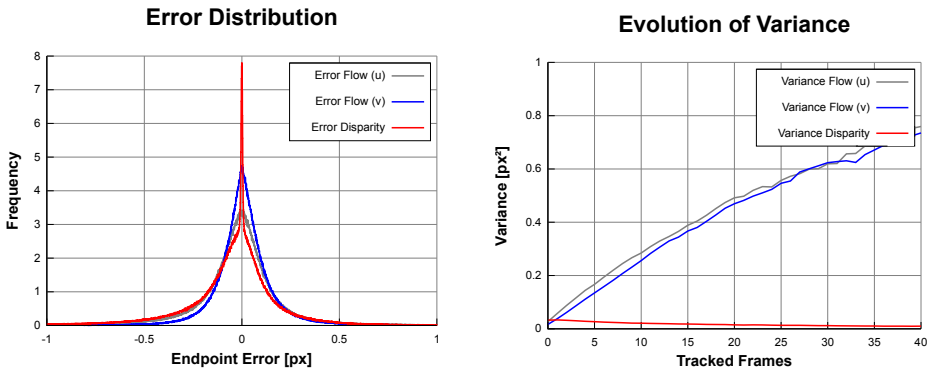

Figure 5.17: Error distribution of the optical flow components u (grey), v (blue) and the disparity (red) estimated by the proposed scene flow algorithm (left), and the evolution of the variance over time (right).

The actual error distributions are shown in the left image of Figure 5.17. Here, the variance of the errors of the optical flow components, estimated using robust statistics, are $\sigma_u^2 = 0.016\,\text{px}^2$ ($\sigma_u = 0.13$ px) and $\sigma_v^2 = 0.010\,\text{px}^2$ ($\sigma_v = 0.10$ px). These are slightly better than the classic KLT, due to the fact that the information of the left and right images contribute to the optical flow. The variance of the error in the disparity is $\sigma_d^2 = 0.017\,\text{px}^2$ ($\sigma_d = 0.13$ px). This is less than half the variance of the correlation stereo algorithm. In the right image of Figure 5.17 the evolution of the variances over the number of tracked frames is shown. Again, the optical flow components are comparable to the classic KLT (see Figure 2.17). However, the variance of the disparity seems to decrease over time, whereas one would expect that the variance remains constant, or even increases slightly. The reason for this process is the implicit epipolar constraint. Features initialised with a wrong disparity are less likely to survive than features with a correctly initialised disparity, and therefore do not contribute to the statistic of

later frames.

Implemented on the GPU, the presented algorithm requires 35 ms per frame tracking a maximum number of 10000 features, and additional 6 ms to detect new features. It is therefore well-suited for real-time applications.

5.3.3 Dense Scene Flow

In [116] Wedel et al. presented the first real-time capable algorithm to estimate a dense optical flow and disparity change field from two pairs of stereo images. This algorithm is based on the total variation approach already described in Section 5.1.2 and is extended by introducing additional constraints for the optical flow in the right images and the disparity in the current image pair. Starting with a given disparity map for the previous stereo image pair, the algorithm minimises the energy

$$\begin{aligned} E \;=\; & \int_{\Omega} \lambda_1 \left| I_l\left(u+\delta u, v+\delta v, t\right) - I_l\left(u, v, t-1\right)\right| \, d\mathbf{x} + \\ & \int_{\Omega} \lambda_2 c\left(\mathbf{x}\right) \left| I_r\left(u_r+\delta u+d', v+\delta v, t\right) - I_r\left(u_r, v, t-1\right)\right| \, d\mathbf{x} + \\ & \int_{\Omega} \lambda_3 c\left(\mathbf{x}\right) \left| I_r\left(u_r+\delta u+\delta d, v+\delta v, t\right) - I_l\left(u+\delta u, v+\delta v, t\right)\right| \, d\mathbf{x} + \\ & \int_{\Omega} \left|\nabla \delta u\left(\mathbf{x}\right)\right| + \left|\nabla \delta v\left(\mathbf{x}\right)\right| + \left|\nabla \delta d\left(\mathbf{x}\right)\right| \, d\mathbf{x} \end{aligned} \tag{5.22}$$

with δu, δv as the optical flow components and δd as the disparity change. The substitution $u_r = u - d_{t-1}$ gives the corresponding position in the right image. The function $c\left(\mathbf{x}\right)$ returns 0 for points with no known disparity, and otherwise 1. The data term exploits the constant brightness assumptions of the optical flow in the left image, the right image, and the current stereo image pair, and the regularisation term enforces smoothness over the optical flow field and the change of disparity.

The algorithm achieves excellent results, but relies on an additional stereo algorithm to provide the disparity map of the previous image pair. This disparity map also defines at which points the obtained information can be used to reconstruct the scene motion, since the 3D points and the 3D motion can only be determined when the originating disparity is known. In addition, errors in the disparity map can have an influence on the obtained optical flow field, since both are coupled by the current image pair.

Therefore, it is desirable to rewrite the algorithm to estimate the optical flow and the disparity field, instead of the disparity change. Following the approach of the previous section, this can be achieved by minimising the

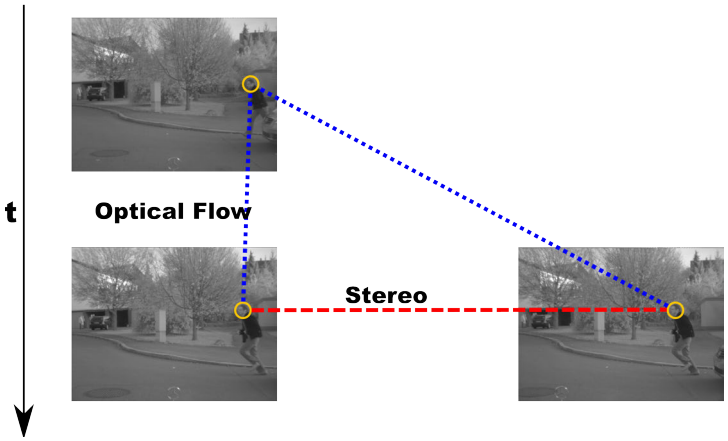

Figure 5.18: Utilised motion and stereo constraints by the proposed dense scene flow algorithm.

energy

$$
\begin{aligned}
E \quad = \quad & \int_{\Omega} \lambda_1 \left| I_l\left(u+\delta u, v+\delta v, t\right) - I_l\left(u, v, t-1\right)\right| \, d\mathbf{x} + \\
& \int_{\Omega} \lambda_2 \left| I_r\left(u+\delta u - d_t, v+\delta v, t\right) - I_l\left(u, v, t-1\right)\right| \, d\mathbf{x} + \\
& \int_{\Omega} \lambda_3 \left| I_r\left(u+\delta u - d_t, v+\delta v, t\right) - I_l\left(u+\delta u, v+\delta v, t\right)\right| \, d\mathbf{x} + \\
& \int_{\Omega} \left|\nabla \delta u\left(\mathbf{x}\right)\right| + \left|\nabla \delta v\left(\mathbf{x}\right)\right| + \left|\nabla d_t\left(\mathbf{x}\right)\right| \, d\mathbf{x}
\end{aligned}
\tag{5.23}
$$

that gives for every point $\mathbf{x}$ in the previous image an optical flow vector $(\delta u,\, \delta v)^\top$ and the disparity in the current image d_t. The utilised constraints are illustrated in Figure 5.18. The energy minimisation problem is then solved using the techniques described in Section 5.1.2.

Like the TV-L^1 stereo algorithm of Section 5.2.2, this approach yields a solution for each pixel, even if it is occluded and therefore not measurable. Following the standard approach of using a forward-backward consistency check, this has to be calculated separately since there are two different paths to check. A good initialisation can be provided by warping the obtained optical flow respifnextchar.. disparity field, and the same engine can be used by disabling the unused computation paths. In practice, it is sufficient to perform the backward verification test using the optical flow from the current right image to the previous left image.

The estimated optical flow field and the obtained disparity map are

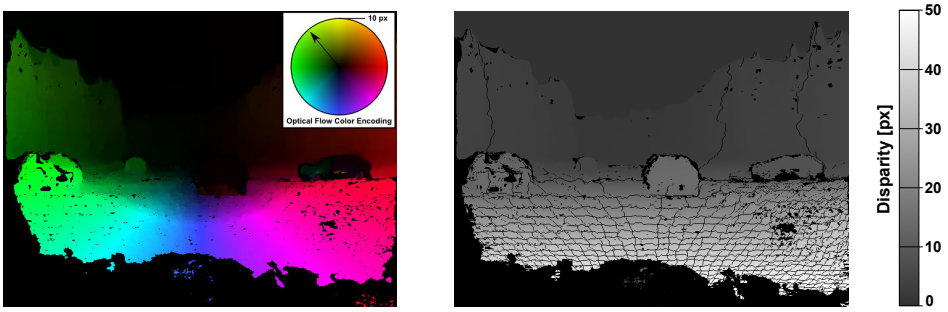

Figure 5.19: Optical flow field and disparity map obtained by the presented dense scene flow algorithm.

shown in Figure 5.19. Here, only pixels surviving the forward-backward consistency check are shown. Since the disparity in the current stereo image pair is calculated only at positions of the previous left image, the shown disparity map results from warping the obtained disparity map by the estimated optical flow field, resulting in the visible cracks. This is no problem for the following processing steps, since they require the disparity information only at positions with a valid optical flow vector.

	Non-occluded	Occluded
AEE	0.198 px (σ=0.686 px)	1.115 px (σ=1.317 px)
AAE	2.22° (σ=6.67°)	8.438° (σ=17.41°)
RMS error	0.714 px	1.726 px
R 1°	41.45 %	53.82 %
R 3°	14.42 %	31.29 %
R 5°	7.35 %	24.72 %
R 0.1 px	48.08 %	94.38 %
R 0.5 px	6.01 %	61.83 %
R 1.0 px	2.49 %	36.85 %
Coverage	53.59 %	22.86 %
Computation time	320 ms	

Table 5.10: Evaluation of the optical flow components of the proposed dense scene flow algorithm.

The results of the evaluation over the ground truth sequence is given in the Tables 5.10 and 5.11. Compared with the dense optical flow (see Table 5.2), the algorithm achieves a smaller average endpoint error, due to the fact that both current images contribute to the flow components. However, the average angular error is slightly larger, meaning that the errors of small flow vectors are larger than for the dense optical flow. Due to the

	Non-occluded	Occluded
AAE	0.169 px (σ=0.479 px)	1.530 px (σ=1.691 px)
RMS error	0.508 px	2.281 px
R 0.5	6.42 %	67.51 %
R 0.75	3.83 %	60.68 %
R 1.0	2.72 %	53.50 %
R 1.5	1.81 %	38.19 %
R 2.0	1.32 %	27.68 %
Coverage	51.88 %	23.27 %

Table 5.11: Evaluation of the Stereo Component of the proposed dense scene flow algorithm.

forward-backward consistency check, the coverage is greatly reduced, since flow vectors are removed from the evaluation if their corresponding vectors from the current right to the previous left image do not match. This reduces - as intended - the coverage in occluded areas, but it is also more sensitive to violations of the constant brightness assumption than the dense optical flow or the TV-L^1 stereo. The stereo accuracy given in Table 5.11 is comparable to the Semi-Global-Matching algorithm using the ZSAD cost function (see Table 5.3).

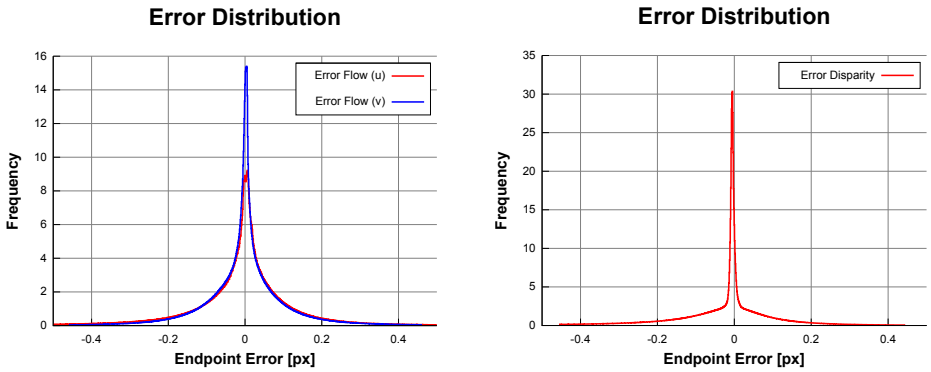

Figure 5.20: Error distribution of the optical flow (left) and disparity (right) components obtained by the dense scene flow algorithm.

The error distributions of each estimated component are given in Figure 5.20. Here, the variances estimated using robust statistics are $\sigma_u^2 = 0.005\,\text{px}^2$ ($\sigma_u = 0.07\,\text{px}$), $\sigma_v^2 = 0.003\,\text{px}^2$ ($\sigma_v = 0.05\,\text{px}$) and $\sigma_d^2 = 0.002\,\text{px}^2$ ($\sigma_d = 0.04\,\text{px}$). Compared to the other algorithms, namely the dense optical flow (see Section 5.1.2) and the TV-L^1 stereo algorithm (see Section 5.2.2), these values show clearly the benefit of combining the optical flow and the stereo analysis in a single approach.

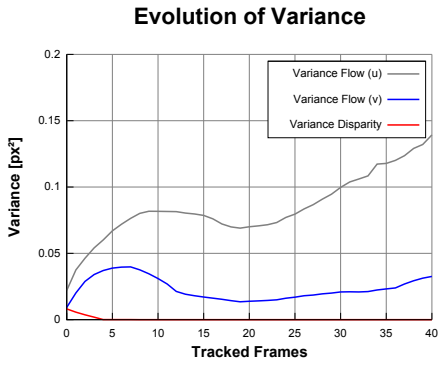

Figure 5.21: Evolution of the variances over time.

To establish tracks from the obtained scene flow, successive optical flow vectors are concatenated over time as described for the dense optical flow in Section 5.1.2. The evolution of the variances over time is shown in Figure 5.21. Here, the variances of the optical flow components increase over the first frames, like for the other presented trackers, suffering from the successive integration of the differential displacement vectors. Since the disparity is not integrated over time but measured each frame directly, it does not show this effect. In fact, the variance of the disparity decreases over time, due to the fact that mismatches are not consistent and eliminated by the consistency check.

The algorithm was implemented on a GPU, and achieves a computation time of about 320 ms. Although this does not qualify to use the algorithm for applications running at frame rates of more than 3 Hz, it shows that a combined estimation of the optical flow and the disparity information is possible and achieves better results than a separate estimation of the components. It must also be noted, that the current implementation does not exploit the estimated information of previous calculations. By using the previous disparity map and optical flow field, it is possible to reduce the number of iterations, and therefore the computation time, significantly.

5.4 Conclusion

In this chapter, various optical flow, stereo and scene flow algorithms were presented and evaluated in detail. To apply the 6D-Vision principle, either a combination of an optical flow and a stereo algorithm, or a scene flow algorithm can be used. The choice of the actual algorithm depends on the required accuracy and robustness, the available hardware and the desired computation time. In Table 5.12, all discussed algorithms are summarised with respect to their accuracy, coverage of non-occluded pixels and speed. Here, the sparse algorithms (KLT, PowerFlow and sparse scene flow) were

	Section	Accuracy	Coverage	Time
KLT	2.3.2	$\sigma_u^2 = 0.016\,\text{px}^2$ $\sigma_v^2 = 0.013\,\text{px}^2$	2.49 %	22 ms (GPU)
PowerFlow	5.1.1	$\sigma_u^2 = 0.059\,\text{px}^2$ $\sigma_v^2 = 0.058\,\text{px}^2$	1.60 %	6 ms (CPU+ECU)
Dense Optical Flow	5.1.2	$\sigma_u^2 = 0.011\,\text{px}^2$ $\sigma_v^2 = 0.008\,\text{px}^2$	98.53 %	33 ms (GPU)
Correlation Stereo	2.3.1	$\sigma_d^2 = 0.049\,\text{px}^2$	79.95 %	140 ms (CPU)
SGM	5.2.1	$\sigma_d^2 = 0.09\,\text{px}^2$	87.56 %	30 ms (FPGA)
TV-L^1 Stereo	5.2.2	$\sigma_d^2 = 0.010\,\text{px}^2$	94.73 %	32 ms (GPU)
Sparse Scene Flow	5.3.2	$\sigma_u^2 = 0.016\,\text{px}^2$ $\sigma_v^2 = 0.010\,\text{px}^2$ $\sigma_d^2 = 0.017\,\text{px}^2$	1.34 %	35 ms (GPU)
Dense Scene Flow	5.3.3	$\sigma_u^2 = 0.005\,\text{px}^2$ $\sigma_v^2 = 0.003\,\text{px}^2$ $\sigma_d^2 = 0.002\,\text{px}^2$	53.59 %	320 ms (GPU)

Table 5.12: Overview of the presented optical flow, stereo and scene flow algorithms.

evaluated with a fixed number of 10000 features, and the given computation times were measured on the indicated hardware.

Based on the presented data, a relative rating of the individual algorithms with respect to their accuracy, robustness against illumination changes and speed is given in Table 5.13. Here, the robustness performance is based on a subjective evaluation of real-world sequences.

Looking at this table, the presented dense scene flow algorithm (see Section 5.3.3) gives outstanding results with respect to accuracy, but its computation time of 320 ms on a GPU hardly qualifies for real-time applications. On the other hand, its sparse counterpart presented in Section 5.3.2 analyses only 10000 points, but achieves the computation on a GPU in only 35 ms. Its accuracy of the optical flow components is comparable to the KLT, and its stereo accuracy is better than the Semi-Global-Matching and Correlation Stereo algorithms.

The dense optical flow and TV-L^1 stereo algorithms yield also good results with respect to accuracy, and both achieve real-time performance on a GPU. However, a combination of these two algorithms on a single GPU results in a computation time of about 65 ms. In addition, the 6D-Vision analysis of such a huge amount of data cannot be handled by a CPU implementation, but has to be performed on the GPU too, thus adding another 12 ms to the total computation time. This results in a frame rate of

	Accuracy	Robustness	Computation Time
KLT	o	+	+
PowerFlow	-	+ +	+ +
Dense Optical Flow	+	-	+
Correlation Stereo	o	+	o
SGM	-	+ +	+
TV-L^1 Stereo	+	-	+
Sparse Scene Flow	+	o	+
Dense Scene Flow	+ +	-	- -

Table 5.13: Relative rating of the presented optical flow, stereo and scene flow algorithms with respect to the accuracy, robustness and computation time. The performance values are + + (very good), + (good), o (acceptable), - (poor) and - - (very poor).

about 12.5 fps. But since the calculation of the optical flow and the stereo can be done in parallel, and the 6D-Vision analysis can be easily split, the full system runs at 25 fps when two or more GPUs are available.

Another option to obtain a dense motion field at 25 fps is to use the dense optical flow algorithm in combination with the SGM algorithm implemented on an FPGA. This requires only one GPU to calculate the optical flow and perform the 6D-Vision analysis.

Looking at the robustness performance given in Table 5.13, the algorithms based on the Census operator, namely the PowerFlow and the SGM algorithms, perform best. This is not surprising, since all other algorithms are based on the constant brightness assumption that is obviously violated by sudden illumination changes. The correlation stereo, as presented in Section 2.3.1, uses the zero-mean sum of squared differences (ZSSD) to compensate for these. However, practical tests reveal that the Census operator performs slightly better than the ZSSD, especially in bad weather situations.

The KLT in its original derivation is quite sensitive to illumination changes, but can be easily modified to use the ZSSD rather than the SSD correlation function, making it robust against illumination changes. This modification is described in more detail in Section 6.1.3.

The presented dense algorithms are not robust against sudden illumination changes or other contamination per se. Although preprocessing methods exist that can improve the robustness, they do not yield satisfying results. For example, using a high-pass filter on the input images eliminates the sensitivity to illumination changes, but also tends to prevent the algorithm from estimating large displacements. A better approach to increase the robustness of this group of algorithms would be to use robust similarity measures in the energy minimisation technique, that is the scope of further research.

Chapter 6

Real-World Results

Figure 6.1: Stereo camera system installed in a Mercedes S-Class.

The real-world video sequences presented throughout this thesis were all taken by a moving observer, i.e., a Mercedes S-Class equipped with a stereo camera system. The infrared-sensitive Bosch Night Vision cameras are attached to the windscreen on a level with the rear-vision mirror, as shown in Figure 6.1. The base width of the stereo camera system is typically about 30 cm, and the cameras have a focal length of 6 respifnextchar.. 12 mm and a resolution of 640×480 respifnextchar.. 1024×332 pixels. The stereo images are captured at a frame rate of 25 fps and processed by a customary personal computer[1] stored in the boot. This setup allows not only to evaluate the algorithms online while driving, but also to record sequences for an offline analysis.

The previously given results of the proposed 6D-Vision principle on real-world data were all taken in good weather conditions. In order to get a thorough picture of the performance of the 6D-Vision algorithm, it is therefore necessary to investigate it more closely when dealing with suboptimal vision conditions. Hence, the qualitative performance of the motion field

[1]The personal computer has an Intel Quad Core 3.0 GHz processor and is equipped with a FPGA card as well as a NVidia GTX 285 graphics card.

estimation in such situations is shown in Section 6.1. In addition, the necessary steps to increase the robustness of the system are explained in detail.

In Section 6.2, the detection and tracking of moving objects based on the estimated motion field is discussed. Here, a simple, yet efficient algorithm is presented which is optimally suited for collision avoidance and mitigation systems. Based on this algorithm, a driver assistance system to avoid collisions with moving pedestrians was realised and is described shortly in Section 6.2.

Throughout this section, the optical flow estimation is performed by the GPU implementation of the Kanade-Lucas-Tomasi tracker (see Section 2.3.2), whereas the hardware implementation of the Semi-Global-Matching algorithm is used for the stereo calculation (see Section 5.2.1). As stated in the previous chapter, this combination allows the efficient analysis of up to 10000 points at 25 fps, including the ego-motion calculation, the 6D-Vision analysis and the object analysis steps.

6.1 Increased Robustness in Bad Weather Situations

6.1.1 Dealing with Images of Low Contrast

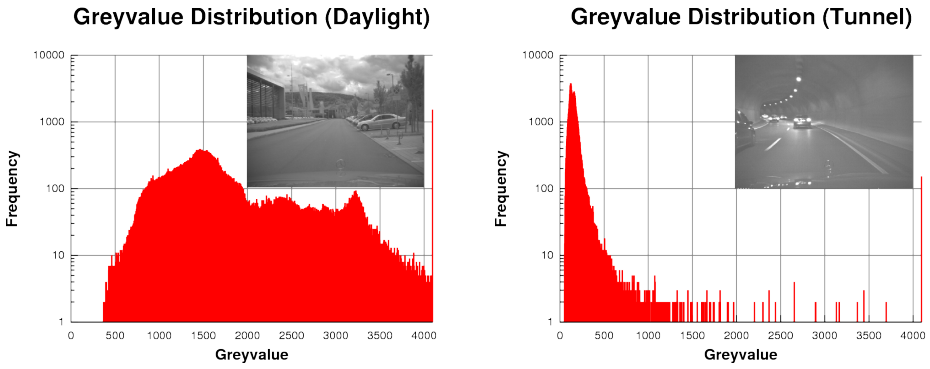

Figure 6.2: Grey value distributions for a daylight and a tunnel scene.

Although the cameras are able to adjust their exposure time, gain and characteristic line to the current illumination situation, even high quality cameras provide dark images with low contrast when the overall illumination is very low. This happens for example at night or when driving through a tunnel. In Figure 6.2, an image of a daylight and a tunnel scene are shown, together with their grey value distributions. For a better illustration, the contrast of the captured images was adjusted by a histogram equalisation. Clearly, the image of the tunnel scene contains mainly small grey values, whereas the daylight scene has a well spread grey value distribution.

To apply the 6D-Vision principle, the stereo and the optical flow information have to be obtained before the motion estimation takes place. Having images of low-contrast, the quality of these two information sources is naturally worse than in daylight scenarios. In order to assure a minimal quality of the optical flow estimation, the KLT tracks only features whose image patch contains a certain amount of structure, measured by the minimal eigenvalue of the structure tensor (see Section 2.3.2). In practice, the feature detector calculates the minimal eigenvalue for each feature candidate, discards all candidates whose value is below a user-defined threshold and sorts the remaining candidates according to their minimal eigenvalue. From this sorted pool of candidates, new features are selected preferring points with high values.

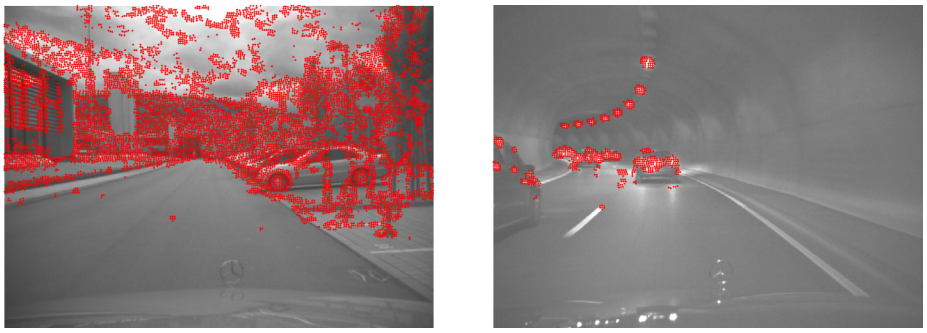

Figure 6.3: Selected features of the KLT using the same threshold for the daylight and tunnel scenario.

The selection of the user-defined threshold does not only assure a certain quality of the optical flow estimation, but has also a severe impact on the processing time of the feature detector, since it defines how many candidates have to be sorted. If it is too small, a large amount of candidates have to be sorted, if it is too high, only a few features will be selected. This dilemma is illustrated in Figure 6.3, showing the selected features using the same threshold for both scenarios. Here, a total of 9187 features is found in the daylight scenario, but only 675 features are selected in the tunnel scenario. Obviously, an optimal threshold selected for a daylight scenario does not yield a satisfying amount of features in the tunnel scenario.

One way to overcome this problem is to determine the optimal threshold, for example by analysing the distribution of the minimum eigenvalues of the feature candidates. However, this leads to computational expensive algorithms that do not outweigh the time required to sort a large amount of candidates when the threshold was chosen too small. Instead of searching for the optimal threshold and refill the feature pool at once, it is therefore better to add a small amount of good features each frame.

This is easily achieved by calculating a non-maxima suppression on the

minimum eigenvalue image, thus taking only the local maximas into account for the later processing steps. By clearing the areas occupied by existing features first, features are assigned to regions of rich structure over time, even if the non-maxima suppression prevents it at a single time step. The window width of the non-maxima suppression is set to the desired minimum distance between neighbouring features, thus assuring that only the absolutely necessary amount of candidates is fed into the following processing steps of the feature detector.

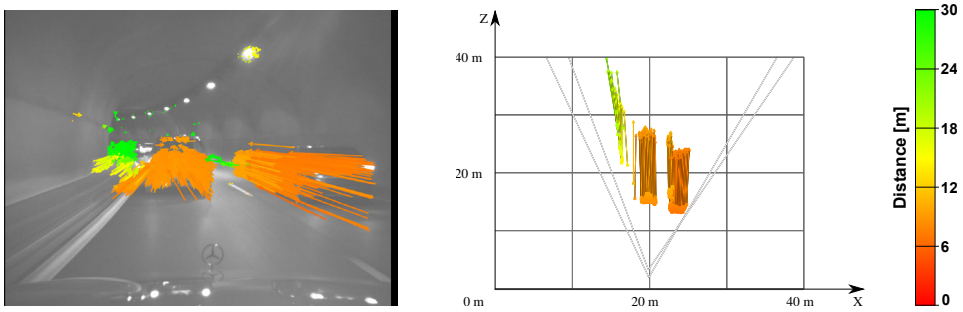

Figure 6.4: Estimated motion field in the tunnel scenario.

The resulting motion field using the modified feature detector for the tunnel scenario is shown in Figure 6.4. Here, the number of features is 3017, about six times as much as with the daylight settings.

6.1.2 Coping with Rainy Weather Conditions

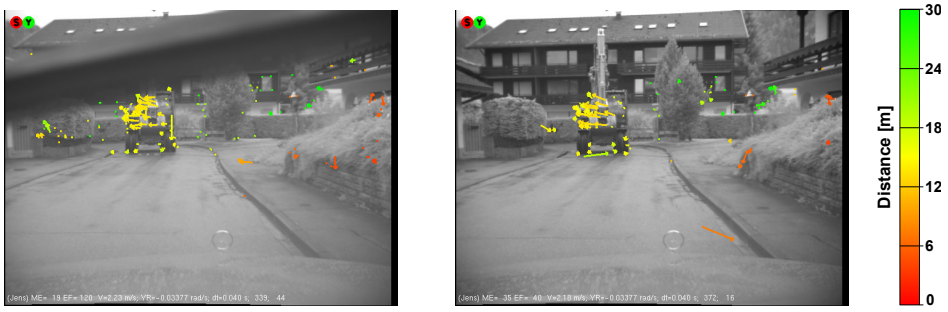

Figure 6.5: Motion field estimation result in the presence of a wiper. All features lost due to the wiper (left) have to be reinitialised, so that no estimation of the motion field is available for the next frames in the occluded image region (right).

In modest rainy weather, the machine vision system is surprisingly less clouded than the human vision system, since the cameras are located close

to the windscreen, thus letting raindrops on the screen act like small lenses. In fact, the major influence on the performance of the 6D-Vision system is the wiper. As it can be seen in Figure 6.5, the wiper does not only remove the raindrops, but also a serious amount of features. Although the system constantly refills its feature pool and therefore assigns immediately new features to the cleared image region, the 6D-Vision estimation takes some frames before reliable motion estimates are available again. If the worst comes to the worst, this leads to the loss of a dangerous moving object and must therefore prevented.

Figure 6.6: Motion field estimation result after skipping the contaminated image containing the wiper.

Due to the installation of the wiper in the car, it is visible in only one of the stereo images at a time. In order to detect such a contamination of the stereo image pair, the difference of the mean grey values between the left and the right image is monitored. If the wiper is present in one of the two images, this value increases significantly, which is easily detected by a simple thresholding technique. In the case of a detection, the image processing is skipped and resumed with the next uncontaminated frame. The result of skipping the frame containing the wiper is given in Figure 6.6. The major advantage of this algorithm is its very short computation time, especially compared to computational expensive change detection methods.

6.1.3 Coping with Illumination Changes

During the evaluation of the 6D-Vision algorithm in the demonstrator vehicle, the major problem turned out to be sudden changes of the exposure time of the camera. Since the cameras were developed for night scenes with the goal of providing sightly images to the driver, neither the exposure settings are available to correct the grey values in a preprocessing step, nor a manual control of the exposure time is possible. The effect of such a change of the apparent illumination on the Kanade-Lucas-Tomasi tracker is shown in Figure 6.7. Although the change is not visible in the grey image, the optical flow estimation of the KLT breaks down due to the violation of the

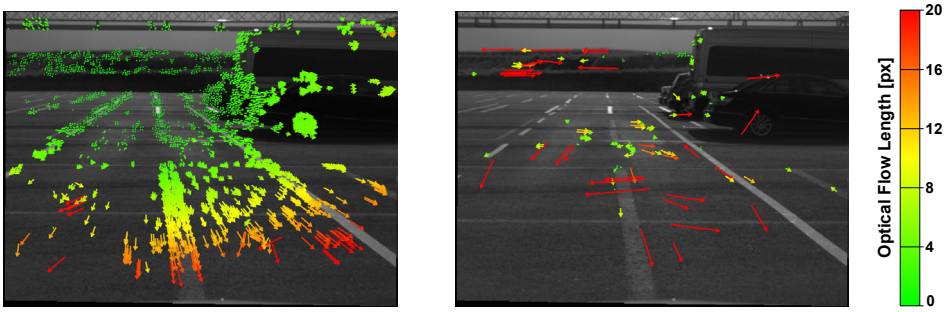

Figure 6.7: Optical flow estimated by the Kanade-Lucas-Tomasi tracker without (left) and with (right) a change of the exposure time of the camera.

constant brightness assumption, and with it the following motion field estimation. Such changes of the camera control are not isolated cases, but can appear multiple times per second.

One way to cope with the problem is to adapt the grey value distributions of both frames under the assumption that a change of the distributions is only due to a changed exposure time. This standard method achieves good results for static scenes, where the assumption is correct. However, for dynamic scenes taken by a moving observer, changes in the grey value distributions are not solely due to changes in the exposure time, and the results of such an adaption are unsatisfactory.

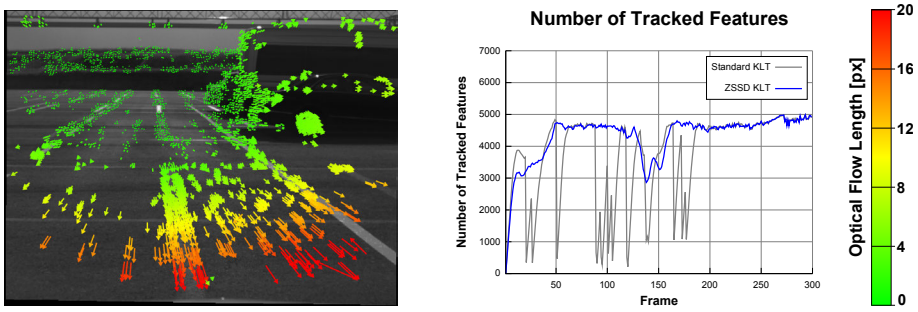

Figure 6.8: Optical flow estimated by the robust Kanade-Lucas-Tomasi tracker during a change of the exposure time of the camera (left). The right image shows the number of tracked features of the classic and the robust version over the complete sequence.

Instead of changing the grey values, it is more efficient to modify the KLT tracker to be robust against such illumination changes. As explained

in Section 2.3.2, the KLT minimises the error function

$$\epsilon = \sum_{W} \left[I\left(\mathbf{x}, t\right) - I\left(h\left(\mathbf{x}, \mathbf{p}\right), t + \delta t\right) \right]^2 \tag{6.1}$$

over the feature window W with respect to the parameter vector $\mathbf{p}$, i.e., the optical flow vector. If the constant brightness assumption is violated, a grey value difference remains that results in a wrong optical flow estimation. To eliminate the influence of that grey value difference, the modified error function

$$\epsilon = \sum_{W} \left[\left(I\left(\mathbf{x}, t\right) - \bar{I}\left(t\right) \right) - \left(I\left(h\left(\mathbf{x}, \mathbf{p}\right), t + \delta t\right) - \bar{I}\left(t + \delta t\right) \right) \right]^2 \tag{6.2}$$

is minimised instead. $\bar{I}\left(t\right)$ respifnextchar.. $\bar{I}\left(t + \delta t\right)$ are the mean grey values of the feature window. This modification can also be interpreted as using the robust ZSSD correlation method instead of the SSD. Since the mean grey values can be computed once for each pixel, the computational burden is nearly neglectable.

The result of the optical flow estimation using this robust variation of the KLT tracker is shown in the left image of Figure 6.8. Compared with Figure 6.7, the optical flow estimation is not affected by the change of the exposure time. This is also illustrated by the evolution of the number of tracked features over the whole sequence, shown in the right image of Figure 6.8. Here, the standard approach suffers from the many adaptions of the exposure control, whereas the robust version is mostly unaffected.

	Non-occluded	Occluded
AEE	0.243 px (σ=0.713 px)	1.076 px (σ=1.580 px)
AAE	3.61° (σ=7.63°)	11.45° (σ=20.04°)
RMS error	0.754 px	1.911 px
R 1°	61.18 %	68.80 %
R 3°	29.66 %	42.94 %
R 5°	17.45 %	33.16 %
R 0.1 px	67.04 %	92.55 %
R 0.5 px	7.60 %	58.36 %
R 1.0 px	2.92 %	34.40 %
Coverage	2.48 %	1.22 %
Computation time (GPU)[2]	18 ms	

Table 6.1: Evaluation of the robust Kanade-Lucas-Tomasi tracker using a maximum of 10000 features.

[2]Time measured on a NVidia GTX 285 graphics card.

The evaluation results on the ground truth sequence are given in Table 6.1. Compared to the results of the standard KLT given in Table 2.3, the robust approach gives slightly worse results.

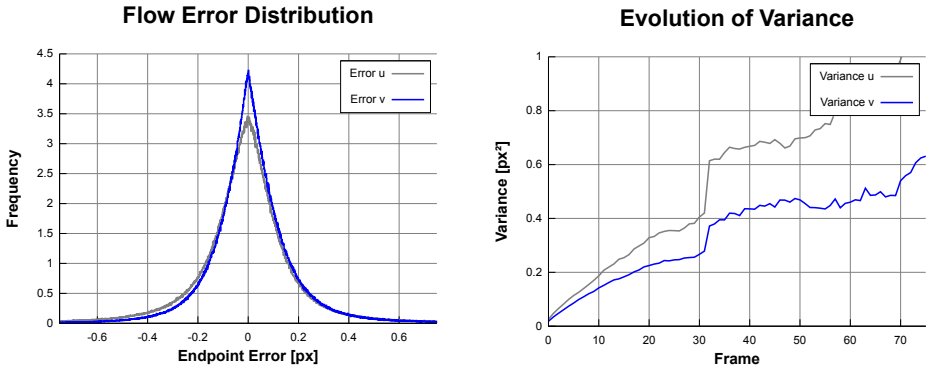

Figure 6.9: Error distribution of the optical flow components$u$ (grey) and v (blue) estimated by the robust Kanade-Lucas-Tomasi tracker (left), and evolution of the variance over multiple frames (right).

However, when looking at the error distribution shown in Figure 6.9 and comparing the error values obtained using robust statistic, namely $\sigma_u^2 = 0.016\,\mathrm{px}^2$ ($\sigma_u = 0.13\,\mathrm{px}$) and $\sigma_v^2 = 0.013\,\mathrm{px}^2$ ($\sigma_v = 0.11\,\mathrm{px}$), the robust version performs just as good as the standard version.

6.2 Object Detection and Tracking

The presented 6D-Vision algorithm in combination with an ego-motion compensation provides the absolute motion field of the observed scene. On the other hand, driver assistance systems that warn the driver in critical situations or even take evasive actions require an object representation to assess the situation. Here, an object is defined as a spacial delimited entity, performing only rigid motions. In order to get from the motion field to an object representation, two steps are necessary: First, moving objects have to be detected, and secondly, found objects have to be tracked over time.

Object Segmentation by Energy Minimisation

The task of detecting moving objects is closely connected to the problem of moving object segmentation, where the focus lies on the determination of the image region belonging to one object. This information is of interest when it is necessary to determine the object boundaries, for example to distinguish between a narrow passing and a collision. A modern algorithm to perform such segmentation tasks is the graph-cut algorithm [118]. Given an energy respifnextchar.. probability for a point to belong to the foreground

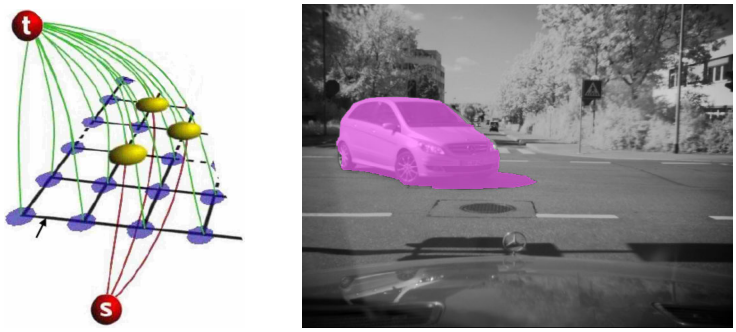

Figure 6.10: Illustration of the graph mapping (left, from [117]) and the result of the graph-cut segmentation (right). Red connections illustrate the graph edges from the source node s, green connections show the edges to the target node t. Black connections illustrate the edges between neighbouring pixels.

respifnextchar.. background, and defining the cost for dividing neighbouring pixels, the algorithm calculates the optimal cut that separates foreground and background, respifnextchar.. moving objects and static image areas. The probability of a point belonging to a moving object is directly obtained from the estimated motion field. However, in order to obtain a pixel precise image segmentation from a sparse 3D motion field, additional information has to be used in the segmentation process. Since object boundaries usually consist with large image gradients, it is reasonable to set the cost for dividing neighbouring pixels inverse proportional to the value of the image gradient [20, 21, 117]. The resulting segmentation is shown in Figure 6.10. Here, the segmentation includes the shadow on the ground as the apparent motion of the shadow is interpreted as a motion in the world.

Although the segmentation results are promising, the graph-cut approach can not be performed in real-time. In addition, the shown segmentation does not distinguish between multiple moving objects. In order to detect moving objects in real-time, the analysis is therefore limited to the estimated motion field, taking into account the requirements of the targeted driver assistance system.

Real-Time Object Detection and Tracking

Depending on the application, an interesting or dangerous object is defined by a characteristic motion. For example, a collision avoidance system defines objects of interest as objects that may cross the way of the vehicle. Since the observed motion field originates from the actual objects, such an analysis can be performed already on the motion field. Assuming constant motion of all participants, the risk of a collision between the vehicle and each motion

Figure 6.11: Result of the risk analysis of the motion field. Red vectors indicate points with a risk of collision in the next 0.5 s, yellow vectors indicate points close to the ego vehicle in the next 0.8 s and green points indicate no risk of collision.

vector during a given time interval in the future is easily calculated. Such an evaluation is shown in Figure 6.11. Here, red vectors indicate points with a high risk of collision for the next 0.5 seconds, whereas points coming close to the vehicle, i.e., less than 1 m in the next 0.8 seconds, are encoded yellow. Points without a risk of collision in the near future are marked green. Although this analysis is performed on the raw motion field information, that is likely to contain outliers, the image illustrates how rich the motion field information is.

Having identified the points of interest, the next step is to build clusters of coherent motion vectors. Here, we assume that all points belonging to an object move at nearly the same speed and in a similar direction. Instead of performing the clustering in the high-dimensional space, it is much faster to separate the analysis of the motion and position components: First, a reference motion vector is identified as the motion vector with the highest frequency in a local neighbourhood. This reference vector is then compared against all available motion vectors in its surrounding neighbourhood, and similar motion vectors are grouped to an object hypothesis. If a sufficiently large number of vectors supports this object hypothesis, a new object entity is established.

To speed-up the neighbourhood search a discretized two-dimensional bucket structure is used, where each bucket contains all motion vectors that are above the ground and lie in the corresponding x, z-area. The comparison of the motion vectors does not use the y-component of the motion vectors, since it gives no usable information in driving scenarios. To account for the uncertainties of the motion vectors, the comparison of two motion vectors uses the Mahalanobis distance instead of the Euclidean distance.

Having identified the objects in one frame, the object tracking step is

responsible to transfer these to the next frame. Since the objects are defined as a set of motion vectors, which already have a correspondence over time, no additional object association step is required. However, the association of each motion vector to the object has to be verified again, taking into account the new motion field data. Since the system refills its feature pool continuously, all unassociated motion vectors are compared to the objects reference vector and eventually assigned to the object.

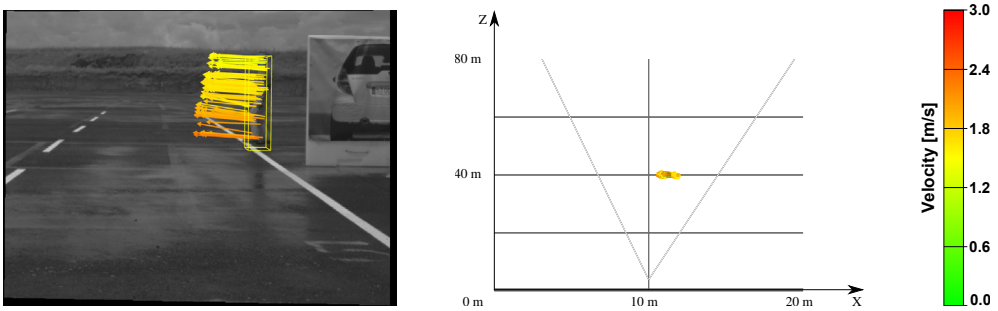

Figure 6.12: Detected object of the object detection and tracking algorithm.

The result of this object detection and tracking algorithm is shown in Figure 6.12. Here, only motion vectors belonging to the identified object are shown. The object is first detected after 3 frames, i.e., when the motion field estimation provides reasonable data.

The described object detection and tracking algorithm is very fast, i.e., about 2 ms on the CPU. However, the limitation to a linear motion model may not be suitable for certain driver assistance systems.

Driver Assistance System for Pedestrian Collision Avoidance

Figure 6.13: Illustration of the autonomous braking (left) and evasion (right) manoeuvre performed by the realised driver assistance system.

Based on the described object detection algorithm, a driver assistance system was realised that is able to react to the thread of an immanent

collision with a moving pedestrian. Depending on the available reaction time, the system either performs an emergency braking, or an autonomous steering manoeuvre if the collision can not be avoided anymore by braking (see Figure 6.13).

The sensor component of the driver assistance system consists of the described object detection based on the 6D-Vision principle, and a pedestrian classification component to assure a reaction on pedestrians only. Since both components share the same computer, the 6D-Vision analysis is only based on 3000 points.

Moving objects detected by the 6D-Vision component are fused with their counterpart of the pedestrian classification in a Kalman Filter, and transferred to the situation analysis. Here, the information is analysed with respect to the risk of a collision and the possible manoeuvres. If possible, an immanent collision is avoided by an emergency braking. However, if the pedestrian was occluded, for example by another car, and is therefore not detected in time, the autonomous steering manoeuvre takes place.

This system was tested thoroughly in different weather conditions and at speeds up to $70\,{}^{\mathrm{km}}/_{\mathrm{h}}$ using a puppet suspended at a traverse, that allows the precise movement of the puppet. Triggered by a light barrier, the pedestrian moves at a constant velocity of $2\,{}^{\mathrm{m}}/_{\mathrm{s}}$ on a collision course. To switch between the two manoeuvres, a movable occlusion is used.

Chapter 7

Conclusion

In this thesis, the novel 6D-Vision principle, that estimates the 3D motion field from the observed image sequence of a stereo camera system, was introduced and investigated in detail (see Chapter 3). This principle fuses the optical flow and stereo information in a probabilistic approach implemented by a Kalman Filter, that allows the real-time estimation of the motion field due to its recursive nature. To overcome the initialisation problem, a filter system was proposed consisting of multiple differently initialised filters, and a unified decision strategy was presented to select the best matching filter. The benefit of the filtering over a direct analysis was illustrated and the algorithm was evaluated using ground truth data. Throughout this thesis, real-world results were presented to demonstrate its practical application.

To estimate the absolute motion field from a moving observer, a new ego-motion algorithm was presented in Chapter 4, that takes advantage of the previously obtained motion estimation as well as available inertial sensor data. Since it is based on a Kalman Filter, it is well suited for implementing more complex vehicle models. Due to the described selection of features, it is robust to measurement noise and errors in the motion field estimation. The algorithm was evaluated on ground truth data, and real-world results were given.

The 6D-Vision principle describes the fusion of the optical flow and stereo information, but is not limited to a particular algorithm to obtain this input information. Throughout this thesis, different optical flow and stereo algorithms were evaluated and compared against each other in a quantitative and qualitative way. Among these, the novel TV-L^1 stereo algorithm gives excellent results with respect to accuracy, and achieves real-time performance at the same time (see Section 5.2.2). The combined analysis of two successive stereo image pairs of the novel sparse and dense scene flow algorithms proved to be superior in terms of accuracy over the individual computation of the optical flow and stereo information (see Section 5.3.2 respifnextchar.. 5.3.3). Although computational expensive, a real-time implementation of

the sparse scene flow algorithm was realised.

The application of the 6D-Vision principle for the detection and tracking of moving objects was given in Chapter 6. Here, the estimated motion field is clustered into objects according to the spacial coherence of the 3D points and the coherence of the 3D motion vectors. A demonstrator car was described in Section 6.2, that performed an autonomous emergency braking respifnextchar.. a steering manoeuvre depending on the observed situation. In addition, various steps were presented to increase the robustness of the motion estimation. In particular, a novel variation of the Kanade-Lucas-Tomasi tracker was presented that estimates the optical flow even in the presence of illumination changes.

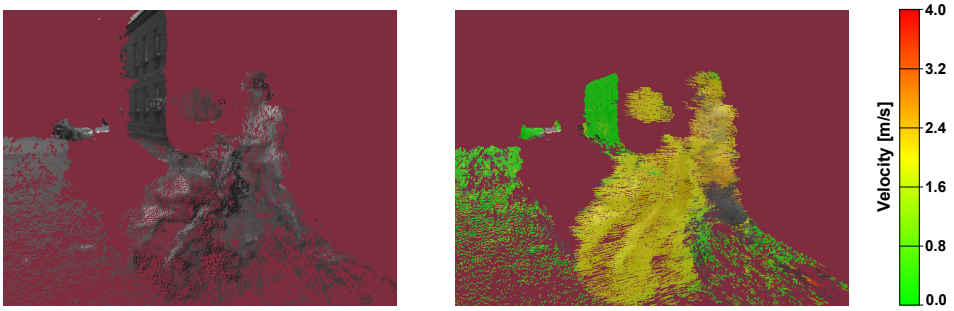

Figure 7.1: 3D reconstruction (left) and dense motion field estimated by the 6D-Vision algorithm (right) for a pedestrian crossing the street. Here, the Semi-Global-Matching and the TV-L^1 optical flow algorithms are used to obtain the dense scene flow data.

The presented system is able to analyse up to 10000 features in real-time, i.e., 25 fps. In order to allow a more precise object segmentation, necessary for example to distinguish between a narrow passing and a collision, future work will focus on implementing the 6D-Vision principle on dense input data. Here, a first result is given in Figure 7.1, showing the estimated motion field based on the presented 6D-Vision algorithm in combination with a dense flow (TV-L^1, see Section 5.1.2) and a dense stereo algorithm (Semi-Global-Matching, see Section 5.2.1).

However, the major drawback of the dense optical flow and scene flow algorithms is currently their sensitivity to violations of the constant brightness assumption. Here, robust similarity measures, like for example the Census operator, have to be implemented and evaluated. Also, the computation time of the presented dense scene flow algorithm must be addressed. One idea is to predict the scene flow from the previous estimate and use it as an initialisation, thus reducing the required number of iterations until convergence for these points.

Having robust dense input information at real-time, the 6D-Vision prin-

ciple can be extended to incorporate not only the information over time, but also over the image. Here it is imaginable to develop an algorithm minimising an error energy based on the 3D motion field with respect to the observed images, while demanding smoothness over the 3D scene.

Since the apparent motion field estimated by the optical flow does not always coincide with the real motion field, virtual motion is estimated for example at shadow borders or areas of reflexions. Such effects are not detectable on a point level, and can only partially be identified when analysing the complete motion field. For example, shadow borders on the ground can be identified by checking the height of the points. However, for a complete understanding of these effects, more complex methods must be developed, which incorporate knowledge about the appearance of potentially moving objects respifnextchar.. static objects.

In addition, modelling specific object motions increases the robustness and the quality of prediction for certain object types. For example, a complex vehicle model proposed by Barth in [119] allows the correct prediction of the movement of the vehicle even in high dynamic turning manoeuvres. Here, the linear motion field estimated by the 6D-Vision algorithm is the basis for establishing object hypothesis, that are then tracked by the vehicle model. Currently, this model is restricted to passenger cars, but can be easily extended to different types of vehicles. By introducing special models for other traffic participants, such as pedestrians or bicyclists, the quality of the estimation and therefore the prediction is improved even further, building a solid basis of decision for upcoming driver assistant systems.

Appendix A

Discretization of a Continuous-Time System

Definition of the Continuous-Time System

A continuous-time system is described by the differential equation system

$$\dot{\mathbf{x}}(t) = \mathbf{A}\,\mathbf{x}(t) + \mathbf{B}\,\mathbf{u}(t) + \boldsymbol{\omega}(t) \tag{A.1}$$

with $\mathbf{x}(t)$ as the state vector, $\mathbf{u}(t)$ as the control input vector, and $\mathbf{A}$ and $\mathbf{B}$ as matrices giving the influence of the state respifnextchar.. control input vector on the change of the state vector. The additive noise vector $\boldsymbol{\omega}(t)$ describes a continuous-time Gaussian white noise, characterised by the covariance matrix $\mathbf{Q}$:

$$\boldsymbol{\omega}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{A.2}$$

A continuous white noise process is also characterised by

$$E\left[\boldsymbol{\omega}(t)\,\boldsymbol{\omega}(\tau)^{\top}\right] = \mathbf{Q}\delta(t-\tau) \tag{A.3}$$

where $\delta(\cdot)$ is the Dirac delta function defined as

$$\delta(x) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{A.4}$$

$$\int_{-\infty}^{\infty} \delta(x)\,dx = 1. \tag{A.5}$$

Equation (A.3) states that the additive noise vector $\boldsymbol{\omega}(t)$ is infinitely correlated with the noise vector $\boldsymbol{\omega}(\tau)$ if $t = \tau$, and has zero correlation for $t \neq \tau$.

Solution of the Differential Equation System

Since

$$\frac{d}{dt}e^{\mathbf{A}t} = \mathbf{A}\,e^{\mathbf{A}t} = e^{\mathbf{A}t}\,\mathbf{A}, \tag{A.6}$$

multiplication of equation (A.1) with $e^{-\mathbf{A}t}$ yields:

$$\begin{aligned} e^{-\mathbf{A}t}\dot{\mathbf{x}}\,(t) &= -\frac{d}{dt}\left(e^{-\mathbf{A}t}\right)\,\mathbf{x}\,(t) + e^{-\mathbf{A}t}\,\mathbf{B}\,\mathbf{u}\,(t) + \\ &\quad e^{-\mathbf{A}t}\,\boldsymbol{\omega}\,(t) && \text{(A.7)} \\ e^{-\mathbf{A}t}\,\dot{\mathbf{x}}\,(t) + \frac{d}{dt}\left(e^{-\mathbf{A}t}\right)\,\mathbf{x}\,(t) &= e^{-\mathbf{A}t}\,\mathbf{B}\,\mathbf{u}\,(t) + e^{-\mathbf{A}t}\,\boldsymbol{\omega}\,(t) && \text{(A.8)} \\ \frac{d}{dt}\left(e^{-\mathbf{A}t}\,\mathbf{x}\,(t)\right) &= e^{-\mathbf{A}t}\,\mathbf{B}\,\mathbf{u}\,(t) + e^{-\mathbf{A}t}\,\boldsymbol{\omega}\,(t)\,. && \text{(A.9)} \end{aligned}$$

Integrating equation (A.9) gives the general solution of the differential equation system as

$$\begin{aligned} e^{-\mathbf{A}t}\,\mathbf{x}\,(t) - e^{0}\,\mathbf{x}\,(0) &= \int_0^t e^{-\mathbf{A}\tau}\,\mathbf{B}\,\mathbf{u}\,(\tau) + e^{-\mathbf{A}\tau}\,\boldsymbol{\omega}\,(\tau)\;d\tau && \text{(A.10)} \\ \mathbf{x}\,(t) &= e^{\mathbf{A}t}\,\mathbf{x}\,(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\,\mathbf{B}\,\mathbf{u}\,(\tau)\;d\tau + \\ &\quad \int_0^t e^{\mathbf{A}(t-\tau)}\,\boldsymbol{\omega}\,(\tau)\;d\tau. && \text{(A.11)} \end{aligned}$$

Derivation of the Corresponding Discrete-Time Model

The discrete-time model describes the state of the process at discrete time steps k. Assuming a constant time interval between the time steps, the state at time step k is

$$\mathbf{x}_k = \mathbf{x}\,(k\,\Delta t) \tag{A.12}$$

with Δt as the time interval.

From equation (A.11) and the fact, that the additive noise error vector

has zero-mean, follows:

$$\mathbf{x}_k = e^{\mathbf{A}k\,\Delta t}\,\mathbf{x}\,(0) + \int_0^{k\,\Delta t} e^{\mathbf{A}(k\,\Delta t-\tau)}\,\mathbf{B}\,\mathbf{u}\,(\tau)\;d\tau \tag{A.13}$$

$$\mathbf{x}_{k+1} = e^{\mathbf{A}(k+1)\Delta t}\,\mathbf{x}\,(0) + \int_0^{(k+1)\Delta t} e^{\mathbf{A}(k\,\Delta t+\Delta t-\tau)}\,\mathbf{B}\,\mathbf{u}\,(\tau)\;d\tau \tag{A.14}$$

$$= e^{\mathbf{A}\Delta t}\,\mathbf{x}_k + \int_{k\Delta t}^{(k+1)\Delta t} e^{\mathbf{A}(k\,\Delta t+\Delta t-\tau)}\,\mathbf{B}\,\mathbf{u}\,(\tau)\;d\tau \tag{A.15}$$

Assuming $\mathbf{u}\,(t)$ is constant during the time interval Δt, and substituting $v = k\,\Delta t + \Delta t - \tau$, the discrete-time model is given as

$$\mathbf{x}_{k+1} = \mathbf{A}_k\,\mathbf{x}_k + \mathbf{B}_k\,\mathbf{u}_k + \boldsymbol{\omega}_k \tag{A.16}$$

with

$$\mathbf{A}_k = e^{\mathbf{A}\Delta t} \tag{A.17}$$

$$\mathbf{B}_k = \int_0^{\Delta t} e^{\mathbf{A}v}\,dv\,\mathbf{B} \tag{A.18}$$

$$\boldsymbol{\omega}_k \sim \mathcal{N}\,(\mathbf{0}, \mathbf{Q}_k) \tag{A.19}$$

The covariance of the state vector at time t is given by

$$E\left[\mathbf{x}\,(t)\,\mathbf{x}\,(t)^\top\right] = E\left[\left(\int_0^t e^{\mathbf{A}(t-\alpha)}\,\boldsymbol{\omega}\,(\alpha)\;d\alpha\right)\left(\int_0^t e^{\mathbf{A}(t-\beta)}\,\boldsymbol{\omega}\,(\beta)\;d\beta\right)^\top\right] \tag{A.20}$$

$$= \int_0^t\int_0^t e^{\mathbf{A}(t-\alpha)}\,E\left[\boldsymbol{\omega}\,(\alpha)\,\boldsymbol{\omega}\,(\beta)^\top\right]\left(e^{\mathbf{A}(t-\beta)}\right)^\top d\alpha\,d\beta \tag{A.21}$$

$$= \int_0^t\int_0^t e^{\mathbf{A}(t-\alpha)}\,\mathbf{Q}\delta\,(\alpha-\beta)\left(e^{\mathbf{A}(t-\beta)}\right)^\top d\alpha\,d\beta \tag{A.22}$$

$$= \int_0^t e^{\mathbf{A}(t-\beta)}\,\mathbf{Q}\,\left(e^{\mathbf{A}(t-\beta)}\right)^\top d\beta. \tag{A.23}$$

Substituting $v = t - \beta$ gives the covariance matrix of the state vector at time t as

$$E\left[\mathbf{x}(t)\,\mathbf{x}(t)^{\top}\right] = \int_0^t e^{\mathbf{A}v}\,\mathbf{Q}\,\left(e^{\mathbf{A}v}\right)^{\top}\,dv \tag{A.24}$$

and the covariance matrix $\mathbf{Q}_k$ of the discrete-time process as

$$\mathbf{Q}_k = \int_0^{\Delta t} e^{\mathbf{A}v}\,\mathbf{Q}\,\left(e^{\mathbf{A}v}\right)^{\top}\,dv \tag{A.25}$$

Interpretation of the Continuous-Time Process Covariance

To compare the relationship between the covariance matrix $\mathbf{Q}$ of the continuous-time process and the covariance matrix $\mathbf{Q}_k$ of the discrete-time process, we consider the simple discrete-time system

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\omega}_k \tag{A.26}$$
$$\mathbf{x}_0 = \mathbf{0} \tag{A.27}$$
$$\boldsymbol{\omega}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)\,. \tag{A.28}$$

The state vector at time step k is therefore

$$\mathbf{x}_k = \boldsymbol{\omega}_0 + \boldsymbol{\omega}_1 + \ldots + \boldsymbol{\omega}_{k-1} \tag{A.29}$$

and the covariance matrix of the state is given as

$$E\left[\mathbf{x}_k\mathbf{x}_k^{\top}\right] = E\left[(\boldsymbol{\omega}_0 + \boldsymbol{\omega}_1 + \ldots + \boldsymbol{\omega}_{k-1})\,(\boldsymbol{\omega}_0 + \boldsymbol{\omega}_1 + \ldots + \boldsymbol{\omega}_{k-1})^{\top}\right] \tag{A.30}$$
$$= E\left[\boldsymbol{\omega}_0\boldsymbol{\omega}_0^{\top}\right] + E\left[\boldsymbol{\omega}_1\boldsymbol{\omega}_1^{\top}\right] + \ldots + E\left[\boldsymbol{\omega}_{k-1}\boldsymbol{\omega}_{k-1}^{\top}\right] \tag{A.31}$$
$$= k\,\mathbf{Q}_k. \tag{A.32}$$

For the corresponding continuous-time system

$$\dot{\mathbf{x}}(t) = \boldsymbol{\omega}(t) \tag{A.33}$$
$$\boldsymbol{\omega}(t) \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{A.34}$$

the covariance matrix of the state at time t is according to equation (A.24)

$$E\left[\mathbf{x}(t)\,\mathbf{x}(t)^{\top}\right] = \mathbf{Q}\,t \tag{A.35}$$

With $t = k\,\Delta t$, we get the relation between the covariance matrix $\mathbf{Q}$ of the continuous-time process and the covariance matrix $\mathbf{Q}_k$ of the discrete-time

process as

$$E\left[\mathbf{x}_k \mathbf{x}_k^\top\right] = E\left[\mathbf{x}\left(k\,\Delta t\right)\,\mathbf{x}^\top\left(k\,\Delta t\right)\right] \tag{A.36}$$

$$k\,\mathbf{Q}_k = \mathbf{Q}\,k\,\Delta t \tag{A.37}$$

$$\mathbf{Q} = \frac{\mathbf{Q}_k}{\Delta t}. \tag{A.38}$$

Appendix B

Derivation of the Kalman Filter

The Kalman Filter estimates the state vector $\mathbf{x}_k$ of the discrete-time linear system model

$$\mathbf{x}_k = \mathbf{A}_k \, \mathbf{x}_{k-1} + \mathbf{B}_k \, \mathbf{u}_k + \boldsymbol{\omega}_k \tag{B.1}$$

from a series of measurements $\mathbf{z}_k$. The measurements are related to the state vector by the linear equation

$$\mathbf{z}_k = \mathbf{H}_k \, \mathbf{x}_k + \boldsymbol{\nu}_k \tag{B.2}$$

also called the measurement model. The additive noise vectors are assumed to be white Gaussian noise according to

$$\boldsymbol{\omega}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right) \tag{B.3}$$

$$\boldsymbol{\nu}_k \sim \mathcal{N}\left(\mathbf{0}, \mathbf{T}_k\right). \tag{B.4}$$

The estimated state vector $\mathbf{x}_k$ deviates from the real, unknown state vector $\check{\mathbf{x}}_k$. This error is modelled as white Gaussian noise according to

$$\left(\check{\mathbf{x}}_k - \mathbf{x}_k\right) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{P}_k\right). \tag{B.5}$$

The Prediction Step

Given an estimated state vector $\mathbf{x}_{k-1}$ and a state covariance matrix $\mathbf{P}_{k-1}$, the state at the next time step is predicted using the system model described by equation (B.1):

$$\widehat{\mathbf{x}}_k = \mathbf{A}_k \, \mathbf{x}_{k-1} + \mathbf{B}_k \, \mathbf{u}_k \tag{B.6}$$

and the predicted state covariance matrix is found as

$$\widehat{\mathbf{P}}_k = \mathbf{A}_k \, \mathbf{P}_{k-1} \, \mathbf{A}_k^\top + \mathbf{Q}_k. \tag{B.7}$$

The Measurement Update

The predicted measurement of the predicted state vector $\widehat{\mathbf{x}}_k$ is derived from the measurement model described by equation (B.2) as

$$\widehat{\mathbf{z}}_k = \mathbf{H}_k \, \widehat{\mathbf{x}}_k \tag{B.8}$$

and its associated covariance matrix is given as

$$\widehat{\mathbf{T}}_k = \mathbf{H}_k \, \widehat{\mathbf{P}}_k \, \mathbf{H}_k^\top . \tag{B.9}$$

The innovation vector

$$\mathbf{s}_k = \mathbf{z}_k - \widehat{\mathbf{z}}_k \tag{B.10}$$

describes the difference between the actual measurement and the predicted measurement. Its covariance matrix is

$$\mathbf{S}_k = \mathbf{T}_k + \widehat{\mathbf{T}}_k \tag{B.11}$$

and is called the innovation covariance matrix.

The new state estimate is derived by updating the predicted state with a weighted and transformed version of the innovation vector:

$$\mathbf{x}_k = \widehat{\mathbf{x}}_k + \mathbf{K}_k \, \mathbf{s}_k \tag{B.12}$$

with $\mathbf{K}_k$ as the Kalman Gain. The Kalman Gain is chosen, so that the resulting state covariance matrix $\mathbf{P}_k$ is minimised.

The Posterior Estimate Covariance Matrix

From the definition of the state covariance matrix in equation (B.5) and the measurement equations described above follows

$$\begin{aligned}
\mathbf{P}_k &= \operatorname{var}\left(\check{\mathbf{x}}_k - \mathbf{x}_k\right) && \text{(B.13)}\\
&= \operatorname{var}\left(\check{\mathbf{x}}_k - (\widehat{\mathbf{x}}_k + \mathbf{K}_k \, \mathbf{s}_k)\right) && \text{(B.14)}\\
&= \operatorname{var}\left(\check{\mathbf{x}}_k - (\widehat{\mathbf{x}}_k + \mathbf{K}_k \, \mathbf{z}_k - \mathbf{K}_k \, \widehat{\mathbf{z}}_k)\right) && \text{(B.15)}\\
&= \operatorname{var}\left(\check{\mathbf{x}}_k - (\widehat{\mathbf{x}}_k + \mathbf{K}_k \, \mathbf{H}_k \, \mathbf{x}_k + \mathbf{K}_k \, \boldsymbol{\nu}_k - \mathbf{K}_k \, \mathbf{H}_k \, \widehat{\mathbf{x}}_k)\right) && \text{(B.16)}\\
&= \operatorname{var}\left((\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k)\left(\check{\mathbf{x}}_k - \widehat{\mathbf{x}}_k\right) - \mathbf{K}_k \, \boldsymbol{\nu}_k\right) . && \text{(B.17)}
\end{aligned}$$

Assuming the measurement noise is independent of the other noise terms, the state covariance matrix can be written as

$$\begin{aligned}
\mathbf{P}_k &= \operatorname{var}\left((\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k)\left(\check{\mathbf{x}}_k - \widehat{\mathbf{x}}_k\right)\right) + \operatorname{var}\left(\mathbf{K}_k \, \boldsymbol{\nu}_k\right) && \text{(B.18)}\\
&= (\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k) \, \operatorname{var}\left(\check{\mathbf{x}}_k - \widehat{\mathbf{x}}_k\right) \, (\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k)^\top + \mathbf{K}_k \operatorname{var}\left(\boldsymbol{\nu}_k\right) \, \mathbf{K}_k^\top && \text{(B.19)}\\
&= (\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k) \, \widehat{\mathbf{P}}_k \, (\mathbf{I} - \mathbf{K}_k \, \mathbf{H}_k)^\top + \mathbf{K}_k \, \mathbf{T}_k \, \mathbf{K}_k^\top && \text{(B.20)}\\
&= \widehat{\mathbf{P}}_k - \widehat{\mathbf{P}}_k \, \mathbf{H}_k^\top \, \mathbf{K}_k^\top - \mathbf{K}_k \, \mathbf{H}_k \, \widehat{\mathbf{P}}_k + \mathbf{K}_k \, \mathbf{S}_k \, \mathbf{K}_k^\top && \text{(B.21)}
\end{aligned}$$

The Optimal Kalman Gain

The optimal Kalman Gain is optimal in the sense that it minimises the expected squared magnitude of the state error $\check{\mathbf{x}}_k - \mathbf{x}_k$, which is equivalent to minimising the trace of the posterior estimate covariance. For the minimum, the matrix derivative must be $\mathbf{0}$

$$\frac{\partial \mathrm{tr}\left(\mathbf{P}_k\right)}{\partial \mathbf{K}_k} = \frac{\partial \mathrm{tr}\left(\widehat{\mathbf{P}}_k - \widehat{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{K}_k^\top - \mathbf{K}_k \mathbf{H}_k \widehat{\mathbf{P}}_k + \mathbf{K}_k \mathbf{S}_k \mathbf{K}_k^\top\right)}{\partial \mathbf{K}_k} \tag{B.22}$$

$$= -\widehat{\mathbf{P}}_k \mathbf{H}_k^\top - \widehat{\mathbf{P}}_k^\top \mathbf{H}_k^\top + \mathbf{K}_k \mathbf{S}_k^\top + \mathbf{K}_k \mathbf{S}_k \tag{B.23}$$

$$= -2\left(\widehat{\mathbf{P}}_k \mathbf{H}_k^\top\right) + 2\mathbf{K}_k \mathbf{S}_k \tag{B.24}$$

$$= \mathbf{0} \tag{B.25}$$

and the optimal Kalman Gain is

$$\mathbf{K}_k = \widehat{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{S}_k^{-1}. \tag{B.26}$$

With the found optimal Kalman Gain, the posterior estimate covariance matrix simplifies to

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \widehat{\mathbf{P}}_k. \tag{B.27}$$

Bibliography

[1] Statistisches Bundesamt, Gruppe VC, *Unfallgeschehen im Strassenverkehr*, Statistisches Bundesamt Wiesbaden, Ed., July 2007. [Online]. Available: http://www.destatis.de

[2] F. Lindner, U. Kressel, and S. Kaelberer, "Robust Recognition of Traffic Signals," in *Proceedings of the Intelligent Vehicles Symposium 2004*, June 2004, pp. 49–53.

[3] N. Barnes, A. Zelinsky, and L. Fletcher, "Real-Time Speed Sign Detection Using the Raidal Symmetry Detector," *Intelligent Transportation Systems*, vol. 9, no. 2, pp. 322–332, June 2008.

[4] B. Cyganek, "Road-Signs Recognition System for Intelligent Vehicles," in *RobVis08*, 2008, pp. 219–233.

[5] R. Risack, P. Klausmann, W. Kruger, and W. Enkelmann, "Robust Lane Recognition Embedded in a Real-Time Driver Assistance System," in *Proceedings of the Intelligent Vehicles Conference 1998*, 1998, pp. 35–40.

[6] H. Badino, R. Mester, T. Vaudrey, and U. Franke, "Stereo-Based Free Space Computation in Complex Traffic Scenarios," in *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpreation*, March 2008, pp. 189–192.

[7] M. C. Martin and H. Moravec, "Robot Evidence Grids," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-06, March 1996.

[8] M. Pollefeys, "Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences," Ph.D. dissertation, Katholieke Univeriteit Leuven, May 1999.

[9] U. Franke and C. Rabe, "Kalman Filter based Depth from Motion with Fast Convergence," in *Proceedings of the 2005 IEEE Intelligent Vehicles Symposium*, 2005, pp. 181–186.

[10] C. Rabe, C. Volmer, and U. Franke, "Kalman Filter based Detection of Obstacles and Lane Boundary in Monocular Image Sequences," in *Proceedings of the 19th workshop on Autonomous Mobile Systems (Autonome Mobile Systeme).* Stuttgart, Germany: Springer, December 2005, pp. 51–58.

[11] J. Klappstein, F. Stein, and U. Franke, "Monocular Motion Detection Using Spatial Constraints in a Unified Manner," in *Proceedings of the Intelligent Vehicles Symposium*, 2006, pp. 261–267.

[12] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-Dimensional Scene Flow," in *Seventh International Conference on Computer Vision (ICCV'99)*, vol. 2, 1999, pp. 722–729.

[13] U. Franke, C. Rabe, H. Badino, and S. Gehrig, "6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception," in *Proceedings of the 27th DAGM Symposium*, 2005, pp. 216–223.

[14] H. Badino, U. Franke, C. Rabe, and S. Gehrig, "Stereo Vision-Based Detection of Moving Objects under Strong Camera Motion," in *Proceedings of the First International Conference on Computer Vision Theory and Applications*, vol. 2, February 2006, pp. 253–260.

[15] C. Rabe, U. Franke, and S. Gehrig, "Fast Detection of Moving Objects in Complex Scenarios," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, June 2007, pp. 398–403.

[16] U. Franke, C. Rabe, and S. K. Gehrig, "Kollisionsvermeidung durch raum-zeitliche Bildanalyse (Collision Avoidance based on Space-Time Image Analysis)," *it - Information Technology*, vol. 49, no. 1, pp. 25–32, 2007.

[17] U. Franke, S. K. Gehrig, H. Badino, and C. Rabe, "Towards Optimal Stereo Analysis of Image Sequences," in *Proceedings of the Second International Workshop on Robot Vision.* Auckland, New Zealand: Springer, February 2008, pp. 43–58.

[18] U. Franke, C. Rabe, S. Gehrig, H. Badino, and A. Barth, "Dynamic Stereo Vision for Intersection Assistance," in *FISITA 2008 World Automotive Congress*, Munich, 2008.

[19] S. Gehrig, C. Rabe, and L. Krüger, "6D Vision Goes Fisheye for Intersection Assistance," in *Proceedings of the 2008 Canadian Conference on Computer and Robot Vision*, May 2008, pp. 34–41.

[20] T. Vaudrey, A. Wedel, C. Rabe, J. Klappstein, and R. Klette, "Evaluation of Moving Object Segmentation Comparing 6D-Vision and

Monocular Motion Constraints," in *Proceedings of the 23rd International Conference on Image and Vision Computing New Zealand*, November 2008.

[21] J. Klappstein, T. Vaudrey, C. Rabe, A. Wedel, and R. Klette, "Moving Object Segmentation Using Optical Flow and Depth Information," in *Proceedings of the 3rd Pacific-Rim Symposium on Image and Video Technology*. Tokyo, Japan: Springer, January 2009, pp. 611–623.

[22] O. Faugeras and Q.-T. Luong, *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and-Some of Their Applications*. The MIT Press, 2001.

[23] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, November 1993.

[24] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[25] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3D Vision: From Images to Geometric Models*, ser. Interdisciplinary Applied Mathematics. Springer, 2004, vol. 26.

[26] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[27] R. G. Willson and S. A. Shafer, "A Perspective Projection Camera Model for Zoom Lenses," in *Proc. Second Conference on Optical 3-D Measurement Techniques*, Zürich, Switzerland, October 1993.

[28] R. G. Willson, "Modeling and Calibration of Automated Zoom Lenses," in *Proceedings of the SPIE 2350: Videometrics III*, October 1994, pp. 170–186.

[29] R. G. Willson, "Modeling and Calibration of Automated Zoom Lenses," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 1994.

[30] R. G. Willson and S. A. Shafer, "What is the Center of the Image?" *Journal of the Optical Society of America A*, vol. 11, no. 11, pp. 2946–2955, November 1994.

[31] D. Brown, "Decentering Distortion of Lenses," *Photometric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[32] J. Heikkila and O. Silven, "A Four-step Camera Calibration Procedure with Implicit Image Correction," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.

[33] T. A. Clarke and J. G. Fryer, "The Development of Camera Calibration Methods and Models," in *The Photogrammetric Record*, vol. 16, no. 91, April 1998, pp. 51–66.

[34] Z. Zhang, "A Flexible New Technique for Camera Calibration," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, 2000, pp. 1330–1334.

[35] L. Krüger, C. Wöhler, A. Würz-Wessel, and F. Stein, "In-Factory Calibration of Multiocular Camera Systems," in *SPIE Photonics Europe (Optical Metrology in Production Engineering)*, Strasbourg, 2004, pp. 126–137.

[36] J.-Y. Bouguet. (2008, June) Camera Calibration Toolbox for Matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

[37] R. Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, pp. 323–344, August 1987.

[38] M. Pollefeys, R. Koch, and L. V. Gool, "A Simple and Efficient Rectification Method for General Motion," in *IEEE International Conference on Computer Vision*, 1999, pp. 496–501.

[39] A. Fusiello, E. Trucco, and A. Verri, "A Compact Algorithm for Rectification of Stereo Pairs," *Machine Vision and Applications*, vol. 12, no. 1, pp. 16–22, 2000.

[40] R. I. Hartley and P. Sturm, "Triangulation," *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, November 1997.

[41] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision*. Prentice Hall, 1998.

[42] G. Sibley, L. Matthies, and G. Sukhatme, "Bias Reduction Filter Convergence for Long Range Stereo," in *Proceedings of the 12th International Symposium of Robotics Research (ISRR'05)*, October 2005, pp. 285–294.

[43] J. R. Schott, *Remote Sensing: The Image Chain Approach*. Oxford University Press, 2007.

[44] S. Chambon and A. Crouzil, "Dense Matching using Correlation: New Measures that are Robust near Occlusions," in *In Proceedings of the British Machine Vision Conference*, vol. 1, September 2003, pp. 143–152.

[45] P. Aschwanden and W. Guggenbuehl, "Experimental Results from a Comparative Study on Correlation-type Registration Algorithms," in *Robust Computer Vision*, 1992, pp. 268–289.

[46] M. Avriel, *Nonlinear Programming: Analysis and Methods.* Dover Publishing, 2003.

[47] J. A. Snyman, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms.* Springer Publishing, 2005.

[48] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *The Quaterly of Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[49] D. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *SIAM Journal of Applied Mathematics*, vol. 11, pp. 431–441, 1963.

[50] C. Zach, T. Pock, and H. Bischof, "A Duality Based Approach for Realtime TV-L1 Optical Flow," in *Proceedings of the 29th DAGM Symposium on Pattern Recognition*, 2007, pp. 214–223.

[51] H. Hirschmüller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. 2, San Diego, CA, USA, June 2005, pp. 807–814.

[52] D. Scharstein, R. Szeliski, and R. Zabih, "A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[53] Middlebury Stereo Vision. [Online]. Available: http://vision.middlebury.edu/stereo/

[54] U. Franke and A. Joos, "Real-Time Stereo Vision for Urban Traffic Scene Understanding," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, October 2000, pp. 273–278.

[55] Q. Tian and M. N. Huhns, "Algorithms for Subpixel Registration," *Computer Vision and Graphical Image Processing*, vol. 35, no. 2, pp. 220–233, 1986.

[56] M. Shimizu and M. Okutomi, "Precise Sub-Pixel Estimation on Area-Based Matching," in *Proceedings of the International Conference on Computer Vision (ICCV 2001)*, 2001, pp. 90–97.

[57] A. Stein, A. Huertas, and L. Matthies, "Attenuating Stereo Pixel-Locking via Affine Window Adaptation," in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 914–921.

[58] J. Barron, D. Fleet, and S. Beauchemin, "Performance of Optical Flow Techniques," in *International Journal of Computer Vision*, vol. 12, no. 1, 1994, pp. 43–77.

[59] B. K. P. Horn, *Robot Vision.* MIT Press, 1986.

[60] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artifical Intelligence*, vol. 17, pp. 185–203, 1981.

[61] H.-H. Nagel and W. Enkelmann, "An Investigation of Smoothness Constraints for the Estimation of Displacement Vector Fields from Image Sequences," *IEEE Transactions PAMI 8*, pp. 565–593, 1986.

[62] A. Wedel, T. Pock, J. Braun, U. Franke, and D. Cremers, "Duality TV-L1 Flow with Fundamental Matrix Prior," in *Proceedings of the 23rd International Conference on Image and Vision Computing New Zealand*, 2008, pp. 1–6.

[63] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of Image Understanding Workshop*, 1981, pp. 121–130.

[64] C. Tomasi and T. Kanade, "Detection and Tracking of Point Features," School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, April 1991.

[65] J. Shi and C. Tomasi, "Good Features to Track," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994.

[66] D. J. Fleet and A. D. Jepson, "Computation of Component Image Velocity from Local Phase Information," *International Journal of Computer Vision*, vol. 5, no. 1, pp. 77–104, 1990.

[67] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," in *Proceedings of the 11th International Conference on Computer Vision*, October 2007, pp. 1–8.

[68] D. L. Hall and J. Llinas, *Handbook of Multisensor Data Fusion.* CRC Press LLC, 2001, pp. 10–30.

[69] L. C. Jain, *Advances in Intelligent Systems for Defence.* World Scientific Pub Co Inc, 2002, p. 394.

[70] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[71] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*, ser. Mathematics in Science and Engineering. New York: Academic Press, 1970, vol. 64.

[72] H. Sorenson, Ed., *Kalman Filtering: Theory and Application.* Los Alamitos, CA: IEEE Press, 1985.

[73] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation.* Academic Press, Inc., New York, 1977.

[74] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," University of North Carolina at Chapel Hill, Department of Computer Science, Tech. Rep. TR 95-041, 1995.

[75] G. Welch and G. Bishop, "An Introduction to the Kalman Filter, SIGGRAPH 2001 course 8," in *Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques, SIGGRAPH 2001.* Los Angeles, CA, USA: ACM Press, Addison-Wesley, August 2001.

[76] A. A. Argyros, P. H. Trahanias, and S. C. Orphanoudakis, "Robust Regression for the Detection of Independent 3D Motion by a Binocular Observer," *Journal of Real Time Imaging*, vol. 4, pp. 125–141, 1998.

[77] L.-P. Morency and T. Darrell, "Stereo Tracking using ICP and Normal Flow Constraint," in *Proceedings of the International Conference on Pattern Regognition*, Quebec, Canada, August 2002.

[78] P. J. Besl and N. D. McKay, "A Method for Registration of 3D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, February 1992.

[79] A. M. Waxman and J. H. Duncan, "Binocular Image Flows: Steps Toward Stereo-Motion Fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 715–729, 1986.

[80] P. J. Kellman and M. K. Kaiser, "Extracting Object Motion during Observer Motion: Combining Constraints from Optic Flow and Binocular Disparity," *JOSA-A*, vol. 12, no. 3, pp. 623–625, March 1995.

[81] S. Mills, "Stereo-Motion Analysis of Image Sequences," in *Proceedings of the first joint Autstralia and New Zealand conference on Digital Image and Vision Computing: Techniques and Applications, DICTA'97 / IVCNZ'97*, December 1997.

[82] S. Heinrich, "Fast Obstacle Detection using Flow/Depth Constraint," in *Proceedings of the IEEE Intelligent Vehicles Symposium 2002*, vol. 2, June 2002, pp. 658– 665.

[83] D. Demirdjian and R. Horaud, "Motion-egomotion Discrimination and Motion Segmentation from Image Pair Streams," in *Computer Vision, Graphics and Image Processing*, vol. 78, no. 1, April 2000, pp. 53–68.

[84] D. Demirdjian and T. Darrel, "Motion Estimation from Disparity Images," MIT Artificial Intelligence Laboratory, Technical Report AI Memo 2001-009, May 2001.

[85] T. Dang, C. Hoffmann, and C. Stiller, "Fusing Optical Flow and Stereo Disparity for Object Tracking," in *Proceedings of the IEEE V. International Conference on Intelligent Transportation Systems*, 2002, pp. 112–117.

[86] A. Suppes, F. Suhling, and M. Hoetter, "Robust Obstacle Detection from Stereoscopic Image Sequences using Kalman Filtering," in *Proceedings of the 23rd DAGM Symposium*, Munich, Germany, September 2001, pp. 385–391.

[87] S. Lee and Y. Kay, "A Kalman Filter Approach for Accurate 3D Motion Estimation from a Sequence of Stereo Images," *Computer Vision and Image Understanding: Image Understanding*, vol. 54, no. 2, pp. 244–258, 1991.

[88] P. Rives, E. Breuil, and B. Espiau, "Recursive Estimation of 3D Features using Optical Flow and Camera Motion," in *Intelligent Autonomous Systems, An International Conference*, North-Holland, 1987, pp. 522–532.

[89] L. Matthies, T. Kanade, and R. Szeliski, "Kalman Filter-based Algorithms for Estimating Depth from Image Sequences," *International Journal of Computer Vision*, vol. 3, no. 3, pp. 209–238, 1989.

[90] Z. Zhang and O. D. Faugeras, "Three-Dimensional Motion Computation and Object Segmentation in a Long Sequence of Stereo Images," Inria, Technical Report RR-1438, July 1991.

[91] B. G. A. Y. and T. A. M., "Simultaneous Stereo-Motion Fusion and 3D Motion Tracking," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP'95*, November 1995, pp. 672–677.

[92] Y. S. Yao and R. Chellappa, "Dynamic Feature Point Tracking in an Image Sequence," in *Proceedings of the 1994 International Conference*

on Pattern Recognition, vol. A, Jerusalem, Israel, October 1994, pp. 654–657.

[93] Y. P. Hung, C. Y. Tang, S. W. Shih, Z. Chen, and W. S. Lin, "A 3D Predictive Visual Tracker for Tracking Multiple Moving Objects with a Stereo Vision System," in *Lecture Notes in Computer Science*, vol. 1024, 1995, pp. 25–32.

[94] C. Y. Tang, Y. P. Hung, S. W. Shih, and Z. Chen, "A 3D Feature-based Tracker for Multiple Object Tracking," in *Proceedings of the National Science Council, ROC(A)*, vol. 23, no. 1, 1999, pp. 151–168.

[95] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, pp. 381–395, June 1981.

[96] I.-K. Jung and S. Lacroix, "High Resolution Terrain Mapping using Low Altitude Aerial Stereo Imagery," in *Proceedings of the 9th IEEE International Conference on Computer Vision ICCV'2003*, Nice, France, October 2003, pp. 946–951.

[97] L. Matthies and S. A. Shafer, "Error Modeling in Stereo Navigation," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 3, pp. 239–248, June 1987.

[98] J. Li and R. Skerjanc, "Stereo and Motion Correspondence in a Sequence of Stereo Images," *Signal Processing: Image Communication*, vol. 5, no. 4, pp. 305–318, 1993.

[99] M. Jenkin and J. K. Tsotsos, "Applying Temporal Constraints to the Dynamic Stereo Problem," *Computer Vision, Graphics and Image Processing*, vol. 33, no. 1, pp. 16–31, 1986.

[100] A. Y.-K. Ho and T.-C. Pong, "Cooperative Fusion of Stereo and Motion," *Pattern Recognition*, vol. 29, no. 1, pp. 121–130, 1996.

[101] Y. Altunbasak, A. M. Tekalp, and G. Bozdagi, "Simultaneous Motion-Disparity Estimation and Segmentation from Stereo," in *Proceedings of the 1994 International Conference on Image Processing*, Austin, Texas, USA, 1994, pp. 73–77.

[102] M. Agrawal, K. Konolige, and L. Iocchi, "Real-Time Detection of Intedependent Motion using Stereo," in *IEEE Workshop on Motion and Video Computing (WACV/MOTION'2005)*, Breckenridge, CO, USA, January 2005, pp. 672–677.

[103] A. Talukder and L. Matthies, "Real-Time Detection of Moving Objects from Moving Vehicles using Dense Stereo and Optical Flow," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, Sendai, Japan, September 2004, pp. 315–320.

[104] T. Lefebvre, H. Bruyninckx, and J. D. Schutter, "Kalman Filters for Nonlinear Systems: A Comparison of Performance," in *International Journal of Control*, vol. 77, no. 7, 2004, pp. 639–653.

[105] H. Badino, "A Robust Approach for Ego-Motion Estimation Using a Mobile Stereo Platform," in *1st International Workshop on Complex Motion (IWCM04)*, Guenzburg, Germany, October 2004.

[106] ——, "Binocular Ego-Motion Estimation for Automotive Applications," Ph.D. dissertation, Goethe University Frankfurt, Frankfurt am Main, Germany, 2008.

[107] A. Zomotor, *Fahrwerktechnik: Fahrverhalten.* Würzburg: Vogel Verlag, 1991.

[108] F. Stein, "Efficient Computation of Optical Flow Using the Census Transform," in *Proceedings of the 26th DAGM Symposium*, 2004, pp. 79–86.

[109] R. Zabih and J. Woodfill, "Non-parametric Local Transforms for Computing Visual Correspondence," in *Proceedings of the Third European Conference on Computer Vision*, May 1994, pp. 151–158.

[110] S. S. Beauchemin and J. L. Barron, "The Computation of Optical Flow," in *ACM Computing Surveys*, vol. 27, no. 3, September 1995, pp. 433–466.

[111] D. J. Fleet and Y. Weiss, "Optical Flow Estimation," in *Handbook of Mathematical Models in Computer Vision*, N. Paragios, Y. Chen, and O. Faugeras, Eds. Springer, 2006, ch. 15, pp. 239–258.

[112] J. Weickert, A. Bruhn, T. Brox, and N. Papenberg, "A Survey on Variational Optic Flow Methods for Small Displacements," in *Mathematical Models for Registration and Applications to Medical Imaging*, O. Scherzer, Ed. Springer, 2006.

[113] J. Weickert, A. Bruhn, N. Papenberg, and T. Brox, "Variational Optic Flow Computation: From Continuous Models to Algorithms," in *Proceedings of the International Workshop on Computer Vision and Image Analysis*, L. Alvares, Ed., Las Palmas de Gran Canaria, 2003.

[114] S. Gehrig, F. Eberli, and T. Meyer, "A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching," in *Proceedings of the 7th*

International Conference on Computer Vision Systems, Liège, Belgium, October 2009.

[115] J. Morat, F. Devernay, and S. Cornou, "Tracking with Stereo-Vision System for Low Speed Following Applications," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, Istanbul, June 2007, pp. 955–961.

[116] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers, "Efficient Dense Scene Flow from Sparse or Dense Stereo Data," in *Proceedings of the 10th European Conference on Computer Vision*, vol. 1. Marseille, France: Springer, October 2008, pp. 739–751.

[117] A. Wedel, A. Meißner, C. Rabe, U. Franke, and D. Cremers, "Detection and Segmentation of Independently Moving Objects from Dense Scene Flow," in *Proceedings of the 7th International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Bonn, Germany: Springer, August 2009, pp. 14–27.

[118] Y. Boykov and V. Kolmogorov, "An Experimental Comparison of Min-cut/Max-flow Algorithms for Energy Minimization in Vision," *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 359–374, 2001.

[119] A. Barth, J. Siegemund, U. Franke, and W. Förstner, "Simultaneous Estimation of Pose and Motion at Highly Dynamic Turn Maneuvers," in *DAGM-Symposium*, 2009, pp. 262–271.