INAUGURAL - DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht - Karls - Universität
Heidelberg

vorgelegt von
Dipl.-Ing. Jens Klappstein
aus Erfurt

Tag der mündlichen Prüfung: 28. Juli 2008

Optical-Flow Based Detection of Moving Objects in Traffic Scenes

Gutachter:	Prof. Dr. Bernd Jähne
	Prof. Dr. Dr. h.c. Hans Georg Bock

Zusammenfassung

Der Verkehr auf den Straßen nimmt immer mehr zu. Dennoch ist die Anzahl der Verkehrstoten kontinuierlich zurückgegangen. Dies liegt vor allem an den passiven Sicherheitssystemen, wie Seitenaufprallschutz oder Airbag, welche in den vergangenen Jahrzehnten entwickelt wurden und heute Standard in allen Neufahrzeugen ist. Zunehmend werden aktive Sicherheitssysteme entwickelt. Sie sind in der Lage Unfälle zu vermeiden oder zumindest abzuschwächen. So werden die Abstandsregeltempomaten (ART), die ursprünglich als Komfortsystem ausgelegt waren, hin zu einem automatischen Notbremssystem entwickelt.

Aktive Sicherheit erfordert Sensoren, die die Umgebung des Fahrzeugs erfassen. Für ART werden Radarsysteme oder Laserscanner eingesetzt. Aber auch Kameras sind interessante Sensoren, da mit ihnen zusätzlich visuelle Informationen wie Verkehrsschilder oder Fahrbahnmarkierungen verarbeitet werden können. Im Straßenverkehr spielen bewegte Objekte (Fahrzeuge, Fahrradfahrer, Fußgänger) eine entscheidende Rolle. Sie zu erkennen ist essentiell für aktive Sicherheitssysteme. Die vorliegende Arbeit setzt sich mit der Detektion von bewegten Objekten mittels einer monokularen Kamera auseinander.

Zur Detektion werden die Bewegungen im Videostrom (optischer Fluss) ausgewertet. Ist die Eigenbewegung und die Lage der Kamera in Bezug zur Straßenebene bekannt, kann die aufgenommene Szene mittels des gemessenen optischen Flusses dreidimensional rekonstruiert werden. In der Arbeit wird ein Überblick über bekannte Algorithmen zur Schätzung der Eigenbewegung gegeben. Darauf aufbauend wird ein geeigneter Algorithmus ausgewählt und um ein Bewegungsmodell erweitert. Letzteres steigert sowohl die Genauigkeit als auch die Robustheit erheblich. Die Lage der Kamera zur Straßenebene wird anhand des optischen Flusses der Straße geschätzt. Hierbei ist zu beachten, dass die Straße zeitweilig wenig texturiert sein kann, was das Messen des optischen Flusses erschwert. Die Folge ist eine ungenaue Schätzung der Kameralage. Ein neuartiger Kalman-Filter Ansatz, welcher die Schätzung der Eigenbewegung und die der Kameralage miteinander verbindet, führt zu deutlich besseren Ergebnissen.

Die 3D Rekonstruktion der aufgenommenen Szene geschieht punktweise für jeden gemessenen optischen Flussvektor. Ein Punkt wird rekonstruiert, indem die Sehstrahlen, gegeben durch den Flussvektor, zum Schnitt gebracht werden. Dies ergibt nur für statische, d.h. nicht bewegte, Punkte ein korrektes Ergebnis. Ferner erfüllen statische Punkte vier Bedingungen: Epipolarbedingung, Trifokalbedingung, Bedingung der positiven Tiefe und der positiven Höhe. Ist mindestens eine Bedingung verletzt, handelt es sich um einen bewegten Punkt. Es wird eine Fehlermetrik entwickelt, welche erstmals alle vier Bedingungen ausnutzt und die Abweichung von den Bedingungen einheitlich und quantitativ beschreibt.

Anhand dieser Fehlermetrik werden die Grenzen der Detektierbarkeit untersucht. Konkret wird gezeigt, dass überholende Objekte sehr gut erkennbar sind, dagegen überholte Objekte (Objekte, die langsamer sind als das Eigenfahrzeug) nur sehr schlecht. Gegenverkehr auf gerader Strecke ist nur unter den zusätzlichen Annahmen, dass die Objekte auf dem Boden stehen und undurchsichtig sind, detektierbar. Eine entsprechende Heuristik wird vorgestellt.

In Summe stellen die entwickelten Algorithmen ein System zur robusten Detektion von fremdbewegten Punkten dar. Auf das Problem der Gruppierung der Punkte zu Objekten wird kurz eingegangen. Es dient als Ausgangspunkt für weitergehende Forschungsaktivitäten.

Abstract

Traffic is increasing continuously. Nevertheless the number of traffic fatalities decreased in the past. One reason for this are the passive safety systems, such as side crash protection or airbag, which have been engineered the last decades and which are standard in today's cars. Active safety systems are increasingly developed. They are able to avoid or at least to mitigate accidents. For example, the adaptive cruise control (ACC) original designed as a comfort system is developed towards an emergency brake system.

Active safety requires sensors perceiving the vehicle environment. ACC uses radar or laser scanner. However, cameras are also interesting sensors as they are capable of processing visual information such as traffic signs or lane markings. In traffic moving objects (cars, bicyclists, pedestrians) play an important role. To perceive them is essential for active safety systems. This thesis deals with the detection of moving objects utilizing a monocular camera.

The detection is based on the motions within the video stream (optical flow). If the ego-motion and the location of the camera with respect to the road plane are known the viewed scene can be 3D reconstructed exploiting the measured optical flow. In this thesis an overview of existing algorithms estimating the ego-motion is given. Based on it a suitable algorithm is selected and extended by a motion model. The latter one considerably increases the accuracy as well as the robustness of the estimate. The location of the camera with respect to the road plane is estimated using the optical flow on the road. The road might be temporary low-textured making it hard to measure the optical flow. Consequently, the road homography estimate will be poor. A novel Kalman filtering approach combining the estimate of the ego-motion and the estimate of the road homography leads to far better results.

The 3D reconstruction of the viewed scene is performed pointwise for each measured optical flow vector. A point is reconstructed through intersection of the viewing rays which are determined by the optical flow vector. This only yields a correct result for static, i.e. non-moving, points. Further, static points fulfill four constraints: epipolar constraint, trifocal constraint, positive depth constraint, and positive height constraint. If at least one constraint is violated the point is moving. For the first time an error metric is developed exploiting all four constraints. It measures the deviation from the constraints quantitatively in a unified manner.

Based on this error metric the detection limits are investigated. It is shown that overtaking objects are detected very well whereas objects being overtaken are detected hardly. Oncoming objects on a straight road are not detected by means of the available constraints. Only if one assumes that these objects are opaque and touch the ground the detection becomes feasible. An appropriate heuristic is introduced.

In conclusion, the developed algorithms are a system to detect moving points robustly. The problem of clustering the detected moving points to objects is outlined. It serves as a starting point for further research activities.

Acknowledgments

First of all, I would like to thank Prof. Dr. Bernd Jähne for supervising my dissertation. He helped me with fruitful discussions and supported me in managing the administrative issues. It was always a pleasure to attend the group seminars where I could exchange experiences with the other members of Bernd's group.

The work described in this thesis was carried out at the department of environment perception at Daimler. I am indebted to my advisor Dr. Fridtjof Stein who initiated this project. He provided an optimal work environment for this thesis with regard to all aspects and the necessary freedom and support for this and many other ideas.

I am very grateful to Dr. Uwe Franke who helped me understanding the Kalman filter, Clemens Rabe for his excellent software support, Hernan Badino for his OpenGL simulation tool, Tobi Vaudrey and Dr. Stefan Gehrig for proof-reading my thesis. Furthermore, I would like to thank Dr. Stefan Gehrig and Andreas Wedel for the productive discussions during my years at Daimler. All my work took place in such a friendly atmosphere which was probably unique.

My special thanks goes to my parents. They are always willing to listen to my problems and support me in all my endeavours. They help me cope during hard times, and keep me grounded in good times.

Contents

List of Figures

Notation

Scalars

x	Scalars are represented by a small letter in normal face.
α	The pitch angle of the camera w.r.t. the road plane
ψ	The yaw angle of the camera w.r.t. the road plane
φ	The roll angle of the camera w.r.t. the road plane
$\Delta\alpha$	The pitch angle from the last to the current image frame
$\Delta\psi$	The yaw angle from the last to the current image frame
$\Delta\varphi$	The roll angle from the last to the current image frame

Vectors

$\mathbf{x}$	Vectors are represented by a small letter in boldface.
$\mathbf{x_w}$	A homogeneous world point
$\mathbf{x_l}$	A homogeneous point in the last image frame
$\mathbf{x_c}$	A homogeneous point in the current image frame
$\mathbf{x_l} \leftrightarrow \mathbf{x_c}$	A corresponding point pair
$\mathbf{l_l}$	A homogeneous line in the last image frame
$\mathbf{l_c}$	A homogeneous line in the current image frame
$\mathbf{t}$	The (inhomogeneous) translation from the last frame to the current frame
$\mathbf{\hat{p}_e}$	Estimated ego-motion

Matrices

$\mathbf{M}$	Matrices are represented by a capital letter in boldface.
$\mathbf{K}$	The 3×3 calibation matrix
$\mathbf{E}$	The 3×3 Essential matrix
$\mathbf{F}$	The 3×3 Fundamental matrix
$\mathbf{H}$	A 3×3 homography matrix or Hessian matrix
$\mathbf{H}_\infty$	The 3×3 infinite homography matrix
$\mathbf{P}$	A 3×4 projection matrix
$\mathbf{R}(x,y,z)$	A 3×3 rotation matrix representing the Euler sequence z,y,x
$\mathbf{R}$	The 3×3 rotation matrix representing the rotation from the last frame to the current frame
$[\mathbf{x}]_\times$	The matrix representation of the cross-product with **x**
$\mathrm{Cov}(\cdot)$	The covariance matrix of a random vector

Operators and Accents

$(\cdot)_i$	The i-th component of a vector
$(\cdot)_{ij}$	The i,j-th element of a matrix
$\lvert \cdot \rvert$	The absolute value of a scalar
$\lVert \cdot \rVert$	The L_2-norm of a vector
$\lVert \cdot \rVert_\infty$	The L_∞-norm of a vector
$[\cdot\lvert\cdot]$	The row-wise concatenation of two vectors / matrices
$\times$	The cross-product of two 3-dimensional vectors
$\cdot^+$	The pseudo-inverse of a matrix
$\rightarrow$	The projection of a point in $\mathbb{P}^n$ onto $\mathbb{R}^n$
$d(\cdot,\cdot)$	The distance in $\mathbb{R}^n$ of two vectors in $\mathbb{P}^n$
$\hat{\cdot}$	An estimated value
$\bar{\cdot}$	A true value (error free)
$\tilde{\cdot}$	An averaged value

Abbreviations

CCA	Connected Component Analysis
CPU	Central Processing Unit
DLT	Direct Linear Transform
DoF	Degree of Freedom
ESP	Electronic Stability Program
FoE	Focus of Expansion
HUMSL	Hessian provided Unconstrained Minimization SoLver
IMO	Independently Moving Object
IMU	Inertial Measurement Unit
IRLS	Iteratively Reweighted Least Squares
KLT	Kanade, Lucas, Tomasi (tracker)
LM	Levenberg-Marquardt (minimization)
MLE	Maximum Likelihood Estimate
RANSAC	RANdom SAmple Consensus
SED	Symmetric Epipolar Distance
SLAM	Simultaneous Localization And Mapping
SSD	Sum of Squared Distances
VGA	Video Graphics Array

Chapter 1

Introduction

1.1 Motivation

Traffic is increasing continuously. Nevertheless the number of traffic fatalities decreased in the past. One reason for this are the safety systems which have been engineered the last decades and which are standard in today's cars.

Passive safety systems such as side crash protection or airbag reduce the potential of an injury in case of an accident. In order to avoid accidents active safety systems have been engineered. For example, the anti-lock brake (ABS) prevents the wheels from being locked, so that the car remains steerable. The electronic stability program (ESP) brakes individual wheels when the car is over-steering or under-steering. Within physical limits the skidding of the car is reduced, and the car remains on course. Investigations showed that many drivers press the brake pedal too moderately when braking in an emergency. The brake assistant system (BAS) assists the driver when performing an emergency brake to obtain maximum deceleration.

ABS, ESP, and BAS process the momentary vehicle state. They do not look into the future and thus cannot avoid accidents if the driver is inattentive. To overcome this, noval driver assistance systems are under development. To look ahead they require sensors perceiving the vehicle environment. Some examples are listed below.

- The adaptive cruise control (ACC) uses a lidar or a radar to obtain the distance and the relative speed of the vehicle ahead. It automatically keeps the right distance to the vehicle. This comfort system typically brakes with a maximum deceleration of $4\frac{m}{s^2}$. If a higher deceleration is needed the driver is just warned acoustically and / or optically.

- The lane departure warning (LDW) detects the lane markings using a camera and warns the driver if he crosses the markings unintended.

- The blind spot monitoring (BSM) detects objects within the blind spot of the rear mirror and warns the driver if any object is present. A camera or a radar provides the necessary information.

The driver assistance systems mentioned above are already offered as an option. In order to avoid accidents they still need the drivers intervention. Variants of above driver assistance systems reacting autonomously are under development.

1.2 Sensors for Driver Assistance Systems

A crucial part of driver assistance systems is the sensor, which must be able to take over parts of the recognition tasks of the human eyes. Although we focus on optical sensor input here, the following list of sensors covers the most popular sensors for driver assistance systems and is provided for completeness. Only one sensor out of this list, the camera, operates passively, i.e. relies solely on reflected, not self-emitted, radiation signals. The other sensors measure the distance by measuring the time of flight of the signal from emission to reception, which is proportional to the distance. Envisioning a world of vehicles equipped with driver assistance systems, interference among similar active sensors might become a problem.

Radar A RAdio Detection And Ranging (RADAR) sensor sends out electro-magnetic waves and senses the incoming reflections. Typical frequencies in the automotive field are 24GHz and 77GHz. The emitted signals are pulse-coded and / or frequency modulated, enabling the concurrent measurement of the distance and the relative speed of objects.

Electrically conducting materials such as iron or aluminium reflect the signal very well. Other materials such as plastic or rubber let pass the rays. Hence, these materials are not detected by radar. Radar works well at day and night. Rain and fog do not deteriorate the signal significantly whereas heavy snowfall causes problems.

One drawback is the limited total opening angle achievable at one time. Even a combination of radar beam signals provides only a limited angular resolution. To get a reasonable opening angle and several signals, the radar beam is usually scanned mechanically or electronically. Scanning is performed very quickly (about 50ms) to avoid skewed range measurements.

Lidar The function of LIght Detection And Ranging (LIDAR) is similar to that of radar. Also electro-magnetic waves are sent out, but the frequency is four magnitues higher, namely 300THz (infrared light). This has an impact on the properties: The signal is strongly focussed, allowing to measure distances and directions highly accurate. The relative speed cannot be measured. The signals are susceptible to rain and fog.

As in case of radar a scanning mechanism (rotating mirror) is required to obtain a reasonable opening angle.

Camera Cameras do not emit any signals. They receive the visible and / or infrared light sent out by light sources such as the sun or street lamps. Cameras produce intensity greyscale or color images that do not deliver direct Euclidean measurements. The images must be processed to obtain these measurements.

Another feature of vision sensors is the ability to detect traffic signs, whereas radar and lidar measure distances which are not discriminative for traffic signs.

With one camera, the distance of static objects can be measured by evaluating the optical flow (image displacements from frame to frame). When using two cameras rigidly mounted with a common field of view (stereo), the distance of moving objects is determined in addition to static objects.

PMD The Photonic Mixer Device (PMD) extends a normal camera by the capability of measuring distances by time-of-flight. It emits pulsed non-focussed infrared light. Each sensor element (pixel) receives the sum of the emitted light and the light from the surroundings. The incoming photons are converted to electrons (charges). A charge swing, synchronized with the emitted light, puts the electrons into two distinct bins. The comparison of the collected charges in both bins yields the phase delay between emitted and received light [Ringbeck *et al.* 07]. The time of flight follows directly from the phase delay.

The PMD technology offers the simultaneous measurement of light intensity and distance. However, larger distances require a high power of emitted light.

1.3 Objectives of this Thesis

We have met several sensors for the perception of the vehicle's environment. Cameras are highly interesting since they are not only able to detect obstacles but also lane markings and traffic signs. The simultaneous applicability of cameras for different functions makes this sensor cost-efficient.

In many applications (ACC, BSM) moving objects play the essential role. With a stereo camera moving objects are reconstructable and thus directly detectable [Franke *et al.* 05]. However, this has a price: the second camera causes additional costs and requires space inside the car. An arbitrary location for the second camera is not possible since it has to be attached rigidly to the first camera.

From these thoughts the question raises as to whether one can detect moving objects using a monocular camera? If yes, how to do so and are there any limits? This dissertation answers these questions.

The most frequent objects in traffic are vehicles (cars and trucks). Many different methods have been developed trying to identify vehicles in monocular images. [Sun *et al.* 06] gives an exhaustive overview. There are knowledge based, appearance based, and motion based methods. The knowledge based methods exploit the symmetry between the left and right half of the vehicle or the fact, that the vehicle creates a shadow in its vicinity. Another method tries to find the corners of the vehicle. Appearance based methods learn the grey-value structures typical for vehicles and recognize these structures online. The motion based method analyzes the optical flow. This method is able to detect arbitrary shaped objects including cyclists and pedestrians and is investigated in this thesis.

1.4 Thesis Overview and Contributions

The thesis is organized following the data processing chain from the image acquisition up to the warning of the driver. Figure 1.1 shows the chain. We now go through the individual blocks. **Contributions of the thesis are written in boldface.** Related work is given in the appropriate chapters.

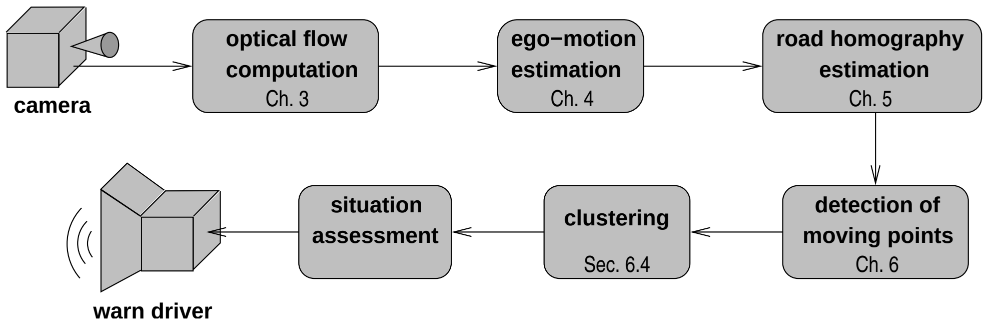


Figure 1.1: Thesis overview and data processing chain. The individual blocks are discussed in the appropriate chapters.

The first step after the image acquisition is the computation of the optical flow. In the literature there are a lot of algorithms computing the optical flow. In chapter 3 the algorithm used in this thesis is explained. This algorithm is designed for the usage within the automotive field.

The detection of moving objects requires the 3D reconstruction of the viewed scene. Note that for a better understanding the detection of moving objects is explained here by means of 3D reconstructed points. The actual algorithm avoids the explicit reconstruction in favour of a reduced computational complexity and a better statistical manageability. The viewed scene is reconstructable if the camera ego-motion from frame to frame is known. The ego-motion can be obtained by two different ways. Firstly, by an inertial measurement unit (IMU) or secondly, by the evaluation of the optical flow. In this thesis the second way is preferred. The computer vision community originated a plethora of algorithms estimating the ego-motion.

In chapter 4 an overview of existing algorithms estimating the ego-motion is given. Based on it a suitable algorithm is selected and extended by a motion model. The latter one considerably increases the accuracy as well as the robustness. The algorithm includes the minimization of a non-linear error function. A slight change of this error function speeds up the minimization. It is shown that the image regions contribute differently to the estimate.

The reconstructed 3D scene lives in the camera coordinate frame. However, it is advantageous if the reconstruction lives in the road coordinate frame, i.e. if the x-z plane coincides with the road plane. Then all 3D points above the road plane have a positive y value. In order to transform the coordinate frame from the camera to the road, the knowledge about the camera location with respect to the road is necessary. The camera location is defined by the normal vector of the road plane and the height of the camera above the road plane. The location itself is a parameter of the road homography. With the road homography one computes the optical flow of a 3D point lying on the road. On the other hand, if the optical flow of several 3D points on the road is given the road homography can be estimated.

In chapter 5 the road homography is estimated. The road might be temporary low-textured making it hard to measure the optical flow. Consequently, the road homography estimate will be poor. A novel Kalman filtering approach combining the estimate of the ego-motion and the estimate of the road homography leads to far better results.

Once the ego-motion and the road homography are known, the moving 3D points can be separated from the static 3D points. This separation relies on the constraints static 3D points fulfill.

In chapter 6 the constraints for static 3D points are named. A novel error metric is introduced combining these constraints in a unified manner. Based on this error metric the detection limits are investigated. It is shown that objects moving anti-parallel with respect to the camera are not detected by means of the available constraints. Only if one assumes that these objects are opaque and touch the ground the detection becomes feasible. An appropriate heuristic is introduced.

The problem of clustering the detected moving points to objects is outlined in section 6.5. A detailed investigation of this problem is beyond the scope of this thesis.

When looking at figure 1.1 one sees that there is a block between the clustering and the final driver warning, the situation assessment. In this block the decision is made whether the detected object constitutes a danger or not. Furthermore, the appropriate reaction is selected. Is it enough to warn the driver (acoustically, optically, or haptically) or should the vehicle be braked? The situation assessment is a research topic of its own and is not addressed in this thesis. The reader is referred to [Hillenbrand 07].

The thesis closes with chapter 7, a summary and outlook. Before we go into detail we address some algebraic and geometric basics, because:

Life is pointless without geometry.

Chapter 2

Mathematical Background

2.1 Projective Geometry

Throughout the thesis we will use a wide range of transformations, including translation, rotation, projection, and other special transformations. Within the Euclidean space these transformations are algebraically expressed in different ways. The translation is represented by a vector-vector addition, the rotation is represented by a matrix-vector multiplication. The projection is performed by a division. Concatenating different types of transformations leads to unaesthetic, not easy to handle, expressions.

The solution of this issue is named projective geometry. It unifies the transformations in such a way that all transformations are expressed by a matrix-vector multiplication. Concatenating transformations means to multiply the matrices of the single transformations. So the overall transformation is described by a single matrix. For example, if we want to translate a point $\mathbf{x}$ by $\mathbf{T}$, then rotate it by $\mathbf{R}$, and finally project it onto the image by $\mathbf{P}$ we can write: $\mathbf{M} = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T}$. The transformed point just computes to $\mathbf{M} \cdot \mathbf{x}$.

Within the projective geometry also the representation of lines and planes is easily done. The next sections discuss the aspects of the projective geometry which are relevant for the thesis. A complete treatment of this topic can be found in several text books, for example [Faugeras & Luong 01], [Hartley & Zisserman 03], or [Ma *et al.* 04].

2.1.1 From Euclidean Space $\mathbb{R}^n$ to Projective Space $\mathbb{P}^n$

Within the n-dimensional Euclidean space a point is uniquely defined by n coordinates. In the projective space the point is extended by one coordinate, i.e. there are $n+1$ coordinates. However, the point still has n degrees of freedom. This means that there is a unique mapping from projective to Euclidean space but not vice versa. The mapping is defined as the central projection through the origin onto the hyper-plane with the $(n+1)$th coordinate being one.

Figure 2.1 illustrates this for the 2-dimensional case. The projective point $(x,y,w)^T \in \mathbb{P}^2$ is associated to the Euclidean point $(x/w, y/w)^T \in \mathbb{R}^2$. The point $(x', y', w')^T$ which is a multiple of $(x,y,w)^T$ is associated to the same Euclidean point. In other words two projective points are equivalent iff they differ only in scale. The point $(0,0,0)^T$ does not exist.

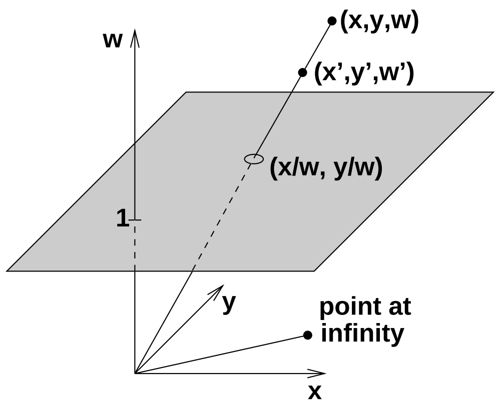


Figure 2.1: The Euclidean space $\mathbb{R}^2$ represented by the plane $w = 1$ is embedded in the projective space $\mathbb{P}^2$. The projective points (x,y,w) and (x',y',w') are both associated to the Euclidean point $(x/w, y/w)$.

Points at infinity The projective space also allows the description of points at infinity which is not possible in the Euclidean space. A short example demonstrates this: Let $\mathbf{x} = (1,1)^T$ be a point in $\mathbb{R}^2$. If the point moves away from the origin $\mathbf{o}$ on the line $\overline{\mathbf{ox}}$ the coordinates grow and grow, and at infinity the coordinates are $\mathbf{x}' = (\infty,\infty)^T$. Unfortunately, all points who went to infinity share the same coordinates. The information from which direction a point was coming is lost. In the projective space this information is preserved. Here $\mathbf{x}$ has the coordinates $\mathbf{x} = (1,1,1)^T$. Going to infinity now means to decrease the last coordinate to zero yielding $\mathbf{x}' = (1,1,0)^T$. All other points at infinity who came from different directions have different coordinates. Due to the unified treatment of finite and infinite points the coordinates of projective points are called *homogeneous coordinates*. Euclidean points have *inhomogeneous coordinates*.

2.1.2 Working with Lines in $\mathbb{P}^2$

In Euclidean space lines can be represented as an equation (known as the Hesse form): $ax + by + c = 0$. In projective space x and y are substituted by x/w and y/w respectively. This leads to the equation: $ax + by + cw = 0$ and in vector notation with $\mathbf{l} = (a,b,c)^T$ and $\mathbf{x} = (x,y,w)^T$:

$$\mathbf{l}^T\mathbf{x} = 0 \tag{2.1}$$

Thus a line is represented by the 3-vector $(a,b,c)^T$ where $(a,b)^T$ corresponds to the normal vector of the line in $\mathbb{R}^2$ and c is the distance to the origin provided that $\left\|(a,b)^T\right\| = 1$. A point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if (2.1) is true. Although $\mathbf{l}$ has three components a line has only two degrees of freedom since (2.1) is immune to an arbitrary scale factor so the two ratios $\{a : b : c\}$ are sufficient to determine a line uniquely.

The line $\mathbf{l}$ joining the points $\mathbf{x_1}$ and $\mathbf{x_2}$ is obtained by:

$$\mathbf{l} = \mathbf{x_1} \times \mathbf{x_2} \tag{2.2}$$

The line at infinity A general point at infinity has the coordinates $(x,y,0)^T$. There is one special line joining all these points. It is $\mathbf{l}_\infty = (0,0,1)^T$. One may check whether these points are part of that line with equation (2.1): $(0,0,1)\,(x,y,0)^T = 0$ which is obviously true. Of course it is impossible to draw this line onto the plane. But when the plane is projected to another plane the line at infinity is mapped to a line with finite coordinates. An example is shown in figure 2.2.

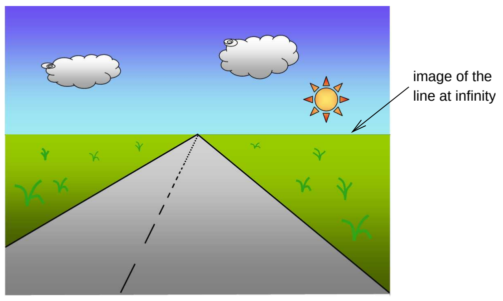


Figure 2.2: The image of the line at infinity. The world plane is projected to the image plane. The line at infinity gets visible.

Duality between points and lines Points and lines are both represented as a 3-vector. In the basic incidence equation for points and lines (see 2.1) the role of both entities is interchangeable since the equation is symmetric: $\mathbf{l}^T\mathbf{x} = \mathbf{x}^T\mathbf{l} = 0$.

The intersection of two lines (2.3) and the line through two points (2.2) are essentially the same, with the roles of points and lines swapped.

Note that this duality only holds in $\mathbb{P}^2$. In $\mathbb{P}^3$ the representation of lines is much more complicated than in $\mathbb{P}^2$. One way of representation are Plücker matrices. However, in this thesis 3D lines are not required. For details refer to the text books mentioned at the beginning of this section. In $\mathbb{P}^3$ there is a duality between points and planes (see section 2.1.4).

Intersection of lines The intersection point $\mathbf{x}$ of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$ is given by:

$$\mathbf{x} = \mathbf{l_1} \times \mathbf{l_2} \tag{2.3}$$

In Euclidean space parallel lines do not have an intersection point. In projective space however they meet at a point at infinity. Consider two lines $\mathbf{l_1} = (a,b,c)^T$ and $\mathbf{l_2} = (a,b,c')^T$ with $c \neq c'$. The intersection point is $\mathbf{l_1} \times \mathbf{l_2} = (c'-c)\,(b,-a,0)^T \cong (b,-a,0)^T$. This point lies on both lines and has infinite large inhomogenous coordinates since the last coordinate is zero. The intersection point only depends on the direction of the lines. A translation (varying c) keeps the point unchanged.

Distance of a point to a line The distance d of a point $\mathbf{x}$ to a line $\mathbf{l}$ in 2D expressed in homogeneous coordinates is:

$$d = \frac{|\mathbf{l}^T \mathbf{x}|}{\|\mathbf{n}\|}$$

where $\mathbf{n}$ is the normal vector of the line (the first two coordinates): $\mathbf{n} = ((\mathbf{l})_1, (\mathbf{l})_2)$. The point $\mathbf{x}$ must be homogenized ($(\mathbf{x})_3 = 1$). Then the equation for d is identical to the Hesse form:

$$d = \frac{(\mathbf{l})_1 \cdot (\mathbf{x})_1 + (\mathbf{l})_2 \cdot (\mathbf{x})_2 + 1 \cdot (\mathbf{l})_3}{\sqrt{(\mathbf{l})_1^2 + (\mathbf{l})_2^2}}$$

and $(\mathbf{l})_3$ is the distance of the line to origin.

Distance between two points The distance vector of the two homogeneous points $(x_1, y_1, w_1)^T$ and $(x_2, y_2, w_2)^T$ is given by:

$$\begin{pmatrix} x_d \\ y_d \\ w_d \end{pmatrix} = \begin{pmatrix} w_2 & 0 & -x_2 \\ 0 & w_2 & -y_2 \\ 0 & 0 & w_2 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ y_1 \\ w_1 \end{pmatrix}$$

Perpendicular line to a given line going through a point A line $\mathbf{m} \in \mathbb{P}^2$ perpendicular to the line $\mathbf{l}$ and going through the point $\mathbf{x}$ not necessarily lying on $\mathbf{l}$ is given by:

$$\mathbf{m} = \begin{pmatrix} (\mathbf{l})_2 \\ -(\mathbf{l})_1 \\ \frac{(\mathbf{l})_1 (\mathbf{x})_2 - (\mathbf{l})_2 (\mathbf{x})_1}{(\mathbf{x})_3} \end{pmatrix}$$

The first two components of the line $\mathbf{l}$ define the direction of the line. The first two components of the perpendicular line $\mathbf{m}$ are built as in the Euclidean space: swap the first two components and put a minus sign to one component.

In matrix notation:

$$\mathbf{m} = \begin{bmatrix} 0 & (\mathbf{x})_3 & 0 \\ -(\mathbf{x})_3 & 0 & 0 \\ (\mathbf{x})_2 & -(\mathbf{x})_1 & 0 \end{bmatrix} \begin{pmatrix} (\mathbf{l})_1 \\ (\mathbf{l})_2 \\ (\mathbf{l})_3 \end{pmatrix}$$

Line given a direction and a point The line $\mathbf{l}$ with the direction $\mathbf{d} = ((\mathbf{d})_1, (\mathbf{d})_2, 0)$ and going through the point $\mathbf{x}$ is:

$$\mathbf{l} = \mathbf{d} \times \mathbf{p}$$

Projection of a point onto a line The point $\mathbf{x_f}$ lying at the foot of the perpendicular to the line $\mathbf{l}$ from the point $\mathbf{x}$ computes to:

$$\mathbf{x_f} = \mathbf{d} \times \mathbf{x} \times \mathbf{l}$$

with $\mathbf{d} = ((\mathbf{l})_1, (\mathbf{l})_2, 0)^T$ the direction of the line perpendicular to $\mathbf{l}$.

In matrix notation:

$$\mathbf{x_f} = \begin{bmatrix} (\mathbf{l})_2^2 & -(\mathbf{l})_1(\mathbf{l})_2 & -(\mathbf{l})_1(\mathbf{l})_3 \\ -(\mathbf{l})_1(\mathbf{l})_2 & (\mathbf{l})_1^2 & -(\mathbf{l})_2(\mathbf{l})_3 \\ 0 & 0 & (\mathbf{l})_1^2 + (\mathbf{l})_2^2 \end{bmatrix} \cdot \mathbf{x}$$

2.1.3 Transformations in $\mathbb{P}^2$

One of the benefits of the projective geometry is that the common transformations are expressed by a matrix-vector multiplication. All transformation matrices are defined up to scale meaning that any arbitrary scaling of the matrix does not change the action of the matrix. To see this consider two transformations of the point $(x, y, w)^T$ once with $\mathbf{M}$ and once with $\lambda\mathbf{M}$, $\lambda \neq 0$:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \mathbf{M} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \rightarrow \begin{pmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \end{pmatrix} \tag{2.4}$$

$$\begin{pmatrix} x'' \\ y'' \\ w'' \end{pmatrix} = \lambda\mathbf{M} \begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} \lambda x' \\ \lambda y' \\ \lambda w' \end{pmatrix} \rightarrow \begin{pmatrix} \frac{\lambda x'}{\lambda w'} \\ \frac{\lambda y'}{\lambda w'} \end{pmatrix} \tag{2.5}$$

The resulting Euclidean point is the same for both transformations since in equation 2.5 the scaling factor λ cancels out.

Due to the duality of points and lines the transformations apply to both entities. But there is an important distinction. If a given transformation $\mathbf{M}$ applies to points:

$$\mathbf{x}' = \mathbf{M}\mathbf{x} \tag{2.6}$$

then lines are transformed according to:

$$\mathbf{l}' = \mathbf{M}^{-T}\mathbf{l} \tag{2.7}$$

The following paragraphs build up a hierarchy of transformations starting with the most specialized ones - translation and rotation - and ending with the most general one - the homography.

Translation A translation in the Euclidean plane using homogeneous coordinates is represented as

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ w \end{pmatrix} \tag{2.8}$$

which is:

$$\begin{pmatrix} x+wt_x \\ y+wt_y \\ w \end{pmatrix} \rightarrow \begin{pmatrix} \frac{x}{w}+t_x \\ \frac{y}{w}+t_y \end{pmatrix} \tag{2.9}$$

This gives exactly the same vector as one gets it if one performs the translation in Euclidean space. There the point is first projected onto $\mathbb{R}^2$: $(x,y,w)^T \rightarrow (x/w,y/w)^T$, and then the translation vector $(t_x,t_y)^T$ is added: $(x/w,y/w)^T + (t_x,t_y)^T = (x/w+t_x,y/w+t_y)^T$.

Rotation A rotation of the coordinate frame about the angle θ using homogeneous coordinates is represented as

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.10}$$

Rotations in the three-dimensional space can be found in appendix A.

Isometry An isometry is composed of a translation, a rotation and a reflection. In $\mathbb{P}^2$ it is represented as:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} \varepsilon\cos\theta & -\sin\theta & t_x \\ \varepsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

with $\varepsilon \in 0,1$. If $\varepsilon = 1$ then the isometry is *orientation-preserving* and is a *Euclidean transformation*. Else if $\varepsilon = -1$ then the isometry reverses orientation. A planar Euclidean transformation can be written more concisely in block form as:

$$\mathbf{x}' = \mathbf{M}_E\mathbf{x} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

This transformation has three degrees of freedom, one for rotation and two for translation. Lengths (distance between two points) and angles (angle between two lines) are invariant. They are not affected by isometries.

Similarity A similarity is an isometry plus an isotropic scaling. In the case of a Euclidean transformation (i.e. no reflection) the similarity has the matrix representation:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

or in block form:

$$\mathbf{x}' = \mathbf{M}_S\mathbf{x} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

This transformation has four degrees of freedom, the scaling accounting for one more degree than a Euclidean transformation. Angles and ratios of lengths are invariant.

Affinity An affinity is a non-singular transformation followed by a translation. In fact it is a similarity plus a perpendicular shear.

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

or in block form:

$$\mathbf{x}' = \mathbf{M}_A\mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

The affine matrix **A** can always be decomposed as:

$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi)\mathbf{D}\mathbf{R}(\phi)$$

where $\mathbf{R}(\theta)$ is the rotation of the Euclidean transformation. The rest represents a shear. To do this first the coordinate frame is rotated into the scaling directions, then $\mathbf{D} = diag(\lambda_1, \lambda_2)$ applies a non-isotropic scaling and finally the coordinate frame is rotated back.

The affinity has two more degrees of freedom than the similarity. These are the angle ϕ and the scaling ratio $\{\lambda_1 : \lambda_2\}$. Parallel lines and ratios of lengths of parallel line segments are invariant to affinities. The line at infinity $\mathbf{l}_\infty$ is fixed under an affine transformation meaning that infinite points stay infinite. However, $\mathbf{l}_\infty$ is not fixed pointwise: Generally a point on $\mathbf{l}_\infty$ is mapped to another point on $\mathbf{l}_\infty$.

Homography The homography is the most general non-singular linear transformation of homogeneous coordinates. It projects points on a plane onto another plane. This is the reason why it is also called planar projective transformation or shortly *projectivity in 2D*. Since we will often meet homographies throughout this thesis the letter **H** is reserved for it. The block form is:

$$\mathbf{x}' = \mathbf{H}\mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \mathbf{x}$$

The homography has eight degrees of freedom according to the nine elements of **H** less one for an arbitrary scale factor. Lengths and angles are not preserved by this transformation, but co-linear points stay co-linear. The cross ratio of four co-linear points is the most fundamental projective invariant. Figure 2.3 shows an example. The cross ratio is given by:

$$\text{Cross} = \frac{d(x_1,x_2) \cdot d(x_3,x_4)}{d(x_1,x_3) \cdot d(x_2,x_4)} \tag{2.11}$$

with $d(\cdot,\cdot)$ representing the distance between two points.

Homographies form a group, i.e. a concatenation of two homographies is a homography. Thus a mapping of image points onto a world plane and from there onto another image is expressed by a single 3×3 matrix. Figure 2.4 illustrates this.

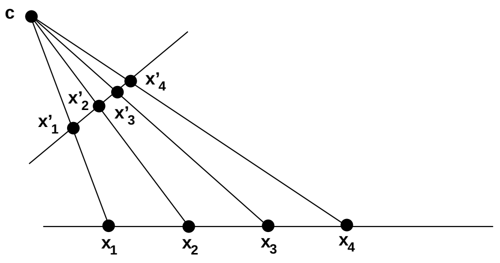


Figure 2.3: A homography transformation of four co-linear points. The cross ratio is invariant under a homography, i.e. $\frac{d(x_1,x_2)\cdot d(x_3,x_4)}{d(x_1,x_3)\cdot d(x_2,x_4)} = \frac{d(x'_1,x'_2)\cdot d(x'_3,x'_4)}{d(x'_1,x'_3)\cdot d(x'_2,x'_4)}$

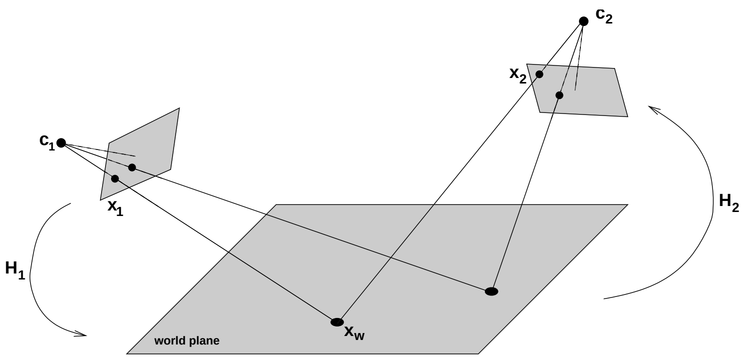


Figure 2.4: Concatenated homography. The homography $\mathbf{H_1}$ maps points from image one onto the world plane. The homography $\mathbf{H_2}$ maps points on the world plane onto image two. The concatenated homography $\mathbf{H_2} \cdot \mathbf{H_1}$ directly maps points from image one onto image two: $\mathbf{x_2} = \mathbf{H_2 H_1 x_1}$

Decomposition of a homography A homography can be decomposed into a chain of transformations, where each matrix in the chain represents a transformation higher in the hierarchy than the previous one.

$$\mathbf{H} = \mathbf{H}_S\mathbf{H}_A\mathbf{H}_P = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{v}^T & v \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \tag{2.12}$$

with $\mathbf{A}$ a non-singular matrix given by $\mathbf{A} = s\mathbf{RK} + \mathbf{tv}^T$ and $\mathbf{K}$ an upper-triangular matrix normalized as $\det(\mathbf{K}) = 1$. This decomposition is valid provided $v \neq 0$, and is unique if s is chosen positive.

2.1.4 Working with Planes in $\mathbb{P}^3$

The representation of planes in $\mathbb{P}^3$ is derived in a similar way as the representation of lines in $\mathbb{P}^2$. A plane in Euclidean space in Hesse form is expressed as: $ax + by + cz + d = 0$. Forming this into homogeneous coordinates and vector notation gives:

$$\pi^T \mathbf{x} = 0 \tag{2.13}$$

where $\pi = (a, b, c, d)^T$ is the plane and $\mathbf{x} = (x, y, z, w)^T$ is a point lying on the plane. π has three degrees of freedom (four minus one for an arbitrary scale factor). The first three components of π correspond to the plane normal of Euclidean geometry.

A plane joining the three points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ is obtained by

$$\begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \mathbf{x}_3^T \end{bmatrix} \pi = \mathbf{0} \tag{2.14}$$

π is the right null-space. It is a one dimensional space if the points are linearly independent (not co-linear). If the points are co-linear then π is a two dimensional null-space and defines a pencil of planes with the line of co-linear points as axis. Instead of calculating the null-space a more convenient direct formula exists which can be found in [Hartley & Zisserman 03].

The plane at infinity As the line at infinity in $\mathbb{P}^2$ the plane at infinity π_∞ in $\mathbb{P}^3$ contains all points lying at infinity. If the space is not projectively distorted the plane at infinity takes the canonical position: $\pi_\infty = (0, 0, 0, 1)^T$, and all points $\mathbf{x}$ with $(\mathbf{x})_4 = 0$ are part of this plane since

$$\begin{pmatrix} (\mathbf{x})_1 \\ (\mathbf{x})_2 \\ (\mathbf{x})_3 \\ 0 \end{pmatrix} \pi_\infty = 0 \tag{2.15}$$

The plane at infinity is a fixed plane under an affinity since infinite points stay infinite. An affine reconstruction of a projectively distorted space (a general homography was applied to the space) is possible if the image of the plane at infinity is known. This reconstruction is done by transforming π_∞ back to its canonical position. The three degrees of freedom of π_∞ measure the projective component of a general homography.

Intersection of planes Having three planes π_1, π_2 and π_3; and stacking the equation 2.13 together gives:

$$\begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{bmatrix} \mathbf{x} = \mathbf{0} \tag{2.16}$$

where $\mathbf{x}$ is a point lying on all planes and thus is the intersection point. With two planes only the null-space $\mathbf{x}$ is two dimensional and defines a pencil of points on the intersection line of the planes.

Duality between points and planes In the two-dimensional case $\mathbb{P}^2$ there is a duality between points and lines (section 2.1.2). Here in $\mathbb{P}^3$ points and planes are dual to each other. Both entities are represented as a 4-vector. The intersection of three planes (2.15) and the plane joining three points (2.16) are essentially the same, with the roles of points and planes swapped.

Note that this duality only holds in $\mathbb{P}^3$. In general points in $\mathbb{P}^n$ are dual to hyper-planes in $\mathbb{P}^n$.

2.1.5 Transformations in $\mathbb{P}^3$

The transformations in $\mathbb{P}^2$ discussed in section 2.1.3 are easily extended to three dimensions. They are not repeated here. But the transformation of planes requires special attention. Due to the duality of points and planes the transformations apply to both entities with a small but important difference: If a given transformation **M** applies to points:

$$\mathbf{x}' = \mathbf{M}\mathbf{x} \tag{2.17}$$

then planes are transformed according to:

$$\pi' = \mathbf{M}^{-T}\pi \tag{2.18}$$

The projection from $\mathbb{P}^3$ to $\mathbb{P}^2$ is deferred until section 2.2.1.

2.2 Image Formation

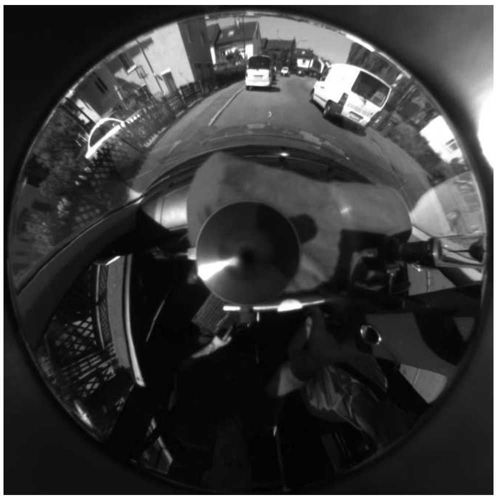
Images are projections of the three-dimensional space onto a two-dimensional space. The latter one can be any free-formed surface.

In order to process the images the light from the 3D scene has to be converted to electrical signals. Technically, this is done by either CCD (charged coupled device) or CMOS (complementary metal oxide semiconductor) sensors. [Litwiller 05] gives a short overview of these two technologies. It is hard to arrange these sensors on surfaces others than a plane. This is the reason why common cameras project the 3D scene onto a plane. Within the plane the sensors are arranged in a rectangular grid. Every single sensor is called a *pixel* standing for picture element.

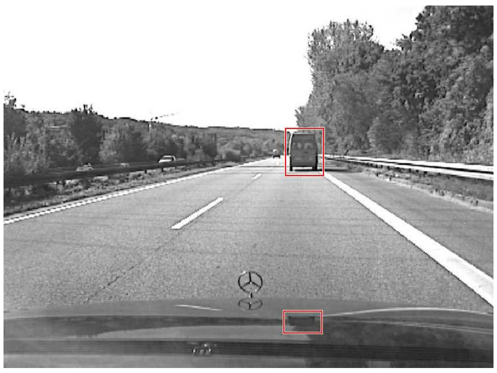
When surfaces other than a plane are desired one uses non-planar mirrors. The 3D scene is first projected onto the mirror, and from there onto the cameras sensor. An example is the hyperbolic mirror enabling an omnidirectional view (360°). An image taken by such a mirror-camera system is shown in figure 2.5a. In [Gehrig 05] two of such systems are employed to reconstruct the 3D scene.

Even free-form surfaces are possible. For instance in [WürzWessel 04] projections of the 3D scene onto the hood of a car are exploited to enable a stereoscopic reconstruction utilizing one camera only. The hood is modelled as a free-form mirror. Figure 2.5b shows an example image. The 3D scene is imaged twice, once directly onto the planar camera sensor and once via the hood.

Throughout this thesis we think of about planar images. When the actual images are taken by a mirror-camera system one may apply a transformation projecting the images virtually onto a plane. This type of transformation is called *rectification* . Note that a real camera does not constitute an exact planar projection due to distortions induced by the lens. Several calibration approaches including distortion models were developed with the aim to measure and to undo this distortion. A well-known approach is the "Camera Calibration Toolbox for Matlab" by Jean-Yves Bouguet
`http://www.vision.caltech.edu/bouguetj/calib_doc/index.html` .



(a)



(b)

Figure 2.5: Types of image surfaces. (a) The 3D scene is projected onto a hyperbolic mirror enabling an omnidirectional view. The camera is mounted near the rear mirror of the car. (courtesy of Stefan K. Gehrig). (b) The vehicle is imaged twice, once directly onto the planar camera sensor and once via the hood, which is modelled as a free-form reflective surface (courtesy of Alexander Würz-Wessel).

2.2.1 Finite Projective Camera

In this section the algebraic description of the projection through the camera center onto a plane, called *central projection* , is discussed. Figure 2.6 illustrates this projection. In a first stage the 3D world point $\mathbf{x} = (x_w, y_w, z_w)^T$ is projected onto the image plane yielding $\mathbf{x}'$. Algebraically this is expressed as follows:

$$\mathbf{x}' = \begin{pmatrix} \frac{f \cdot x_w}{z_w} \\ \frac{f \cdot y_w}{z_w} \end{pmatrix} \tag{2.19}$$

where f is the distance of the image plane to the camera center, also called *focal length* . Thanks to the projective geometry equation 2.19 can be written as a matrix-vector multiplication:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \underbrace{\begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \cdot \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{2.20}$$

The 3×4 *projection matrix* $\mathbf{P}$ is not invertable. It is apparent that a projection comes with a loss of information. Once the world point is projected it is impossible to reconstruct it from the image

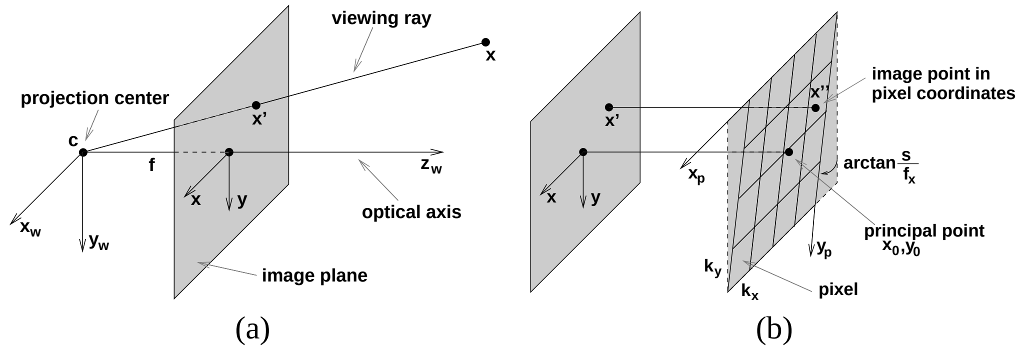


Figure 2.6: Pinhole camera model with pixels. The 3D world point $\mathbf{x}$ is projected onto the image plane yielding $\mathbf{x}'$ (a). The camera center is placed at the coordinate origin. The image plane is parallel to the x_w,y_w plane and lies at $x_z = f$. After the projection the point is transformed to pixel coordinates (b). The origin is the top-left corner. The width of a pixel is k_x, the height is k_y. The point where the optical axis meets the pixel coordinate frame is called principal point.

point. Nevertheless, one may compute the pseudo-inverse of $\mathbf{P}$:

$$\mathbf{P}^+ = \begin{bmatrix} 1/f & & \\ & 1/f & \\ & & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.21}$$

One verifies that $\mathbf{P}^+\mathbf{x} = (x,y,z,0)^T$ for any image point $\mathbf{x}$, i.e. the resulting world point lies at infinity.

After the projection the image point is transformed to pixel coordinates (fig. 2.6b). The origin of the pixel coordinate frame is the top-left corner. Every single pixel has a width of k_x and a height of k_y. The coordinate axes x_p and y_p need not stand perpendicular. The skew parameter s accounts for this. For most normal cameras s will be zero. The point where the optical axis meets the pixel coordinate frame is called *principal point* and has the coordinates $(x_0,y_0)^T$. The transformation to pixel coordinates then reads:

$$\mathbf{x}'' = \begin{pmatrix} x'' \\ y'' \\ w'' \end{pmatrix} = \begin{bmatrix} \frac{1}{k_x} & \frac{s}{f} & x_0 \\ & \frac{1}{k_y} & y_0 \\ & & 1 \end{bmatrix} \cdot \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} \tag{2.22}$$

With the focal length expressed in units of pixel width $f_x = f/k_x$ and pixel height $f_y = f/k_y$ the overall transformation is

$$\mathbf{x}'' = \begin{pmatrix} x'' \\ y'' \\ w'' \end{pmatrix} = \left[\underbrace{\begin{matrix} f_x & s & x_0 \\ & f_y & y_0 \\ & & 1 \end{matrix}}_{\mathbf{K}} \;\; \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \right] \cdot \begin{pmatrix} x_w \\ y_y \\ z_w \\ 1 \end{pmatrix} \tag{2.23}$$

The transformation matrix may also be written as $\mathbf{K}[\mathbf{I}|\mathbf{0}]$ with $\mathbf{I}$ the 3×3 identity matrix. The 3×3 matrix $\mathbf{K}$ captures all intrinsic camera parameters and is called *calibration matrix*. If $\mathbf{K}$ is known one says that the camera is *calibrated* otherwise it is *uncalibrated*. In the field of the industrial image processing (as well as in driver assistance systems) the utilized cameras are known in advance so they can be calibrated before use. Throughout the thesis the camera is considered calibrated.

Sometimes it is not practical to work with pixel coordinates. One can undo the effect of $\mathbf{K}$ through multiplication of the pixel coordinates by the inverse of $\mathbf{K}$: $\mathbf{x}' = \mathbf{K}^{-1}\mathbf{x}''$. For the computation of $\mathbf{K}^{-1}$ see appendix B.1. The coordinates represented by $\mathbf{x}'$ then are *normalized image coordinates*.

Camera rotation and translation So far, the camera center coincided with the origin of the world coordinate frame. This will be unlikely in real life. Instead the camera will be rotated and translated with respect to the world coordinate frame. In order to apply the projection (equation 2.23) the world coordinate frame first has to be transformed into the camera coordinate frame. This is done by an Euclidean transformation:

$$\mathbf{x_c} = \begin{bmatrix} \mathbf{R} & -\mathbf{Rt} \\ \mathbf{0}^T & 1 \end{bmatrix} \cdot \mathbf{x_w} \tag{2.24}$$

The 3×3 rotation matrix $\mathbf{R}$ performs the rotation. The translation is performed by $-\mathbf{Rt}$ where $\mathbf{t}$ is the (inhomogeneous) location of the camera center in the world coordinate frame. The point $\mathbf{x_c}$ is then projected using equation 2.23. Combining both transformations yields:

$$\mathbf{x}'' = \mathbf{KR}[\mathbf{I}| - \mathbf{t}] \cdot \mathbf{x_w} \tag{2.25}$$

This is the algebraic description of a *finite projective camera* with $\mathbf{P} = \mathbf{KR}[\mathbf{I}| - \mathbf{t}]$ the projection matrix. It will be used within this thesis.

Pseudo-inverse The pseudo-inverse of $\mathbf{P}$ maps image points onto a certain world plane π which is derived now. An inverse projected image point $\mathbf{x}$ lies onto π if:

$$\left(\mathbf{P}^{+}\mathbf{x}\right)^{T} \pi = \mathbf{x}^{T}\mathbf{P}^{+T}\pi = 0 \tag{2.26}$$

From section B.2 we know that $\mathbf{P}^{+} = [\mathbf{I}| - \mathbf{t}]^{+}\mathbf{R}^{T}\mathbf{K}^{-1}$. Putting this into equation 2.26 yields

$$\mathbf{x}^{T}\mathbf{K}^{-T}\mathbf{R}[\mathbf{I}| - \mathbf{t}]^{+T}\pi = 0 \tag{2.27}$$

Setting $\pi = (\mathbf{t}^{T}|1)^{T}$ which is the right null-space of $[\mathbf{I}| - \mathbf{t}]^{+T}$ solves equation 2.27. Thus inverse projected image points lie on the world plane $\pi = (\mathbf{t}^{T}|1)^{T}$. If there is no translation between the camera and the worlds origin the points lie on the plane at infinity.

2.2.2 Affine Cameras

For the sake of completeness specialized cameras are discussed in this section. The reason why these cameras are not applicable in driver assistence systems is also given.

The finite projective camera in general does not map parallel lines in the world to parallel lines in the image. This perspective distortion depends on the distance of the camera to the object which is looked at, and on the depth variation of the object. Parallel lines in the world become more and more parallel in the image with increasing distance and decreasing the objects depth variation. This is due to the fact that the viewing rays become more and more parallel. For very large distances the viewing rays can be considered parallel. The central projection transforms to a parallel projection leading to the *affine camera* .

Algebraically an affine camera has a projection matrix **P** in which the last row is of the form $(0,0,0,1)^T$. From this there follow the properties of an affine camera:

- The camera center lies at infinity.
- Parallelism is preserved.
- Points at infinity are mapped to points at infinity.
- The principal point is not defined.

There are some important specializations of the affine camera. Figure 2.7 shows how these specializations act on a world point. They are discussed now starting with the basic operation of parallel projection. More general cases of parallel projection will follow.

Orthographic projection A parallel projection along the $\mathbf{z_w}$ axis is called orthographic. This type of projection ignores the depth of an object. Two identical objects placed at different depths have identical images. Actually, one would expect that the size of the imaged object is smaller for larger depths. The weak-perspective projection, discussed next, accounts for that expectation. The orthographic projection is represented by the matrix:

$$\mathbf{P_o} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

Weak-perspective projection In the weak-perspective projection the 3D scene is "flattened" to a fronto-parallel plane (a plane parallel to the image plane). The depth of that plane is the average depth z_{avg} of the 3D scene. It means the world points are projected orthographically onto that plane. From there the points are projected perspectively onto the image plane, which in this case is nothing else than a simple scaling by f/z_{avg}. Within each weak-perspective view, there is still no variation of reprojection size with the distance. However, the scale can change with each view, as opposed to the orthographic projection. This makes it possible to account

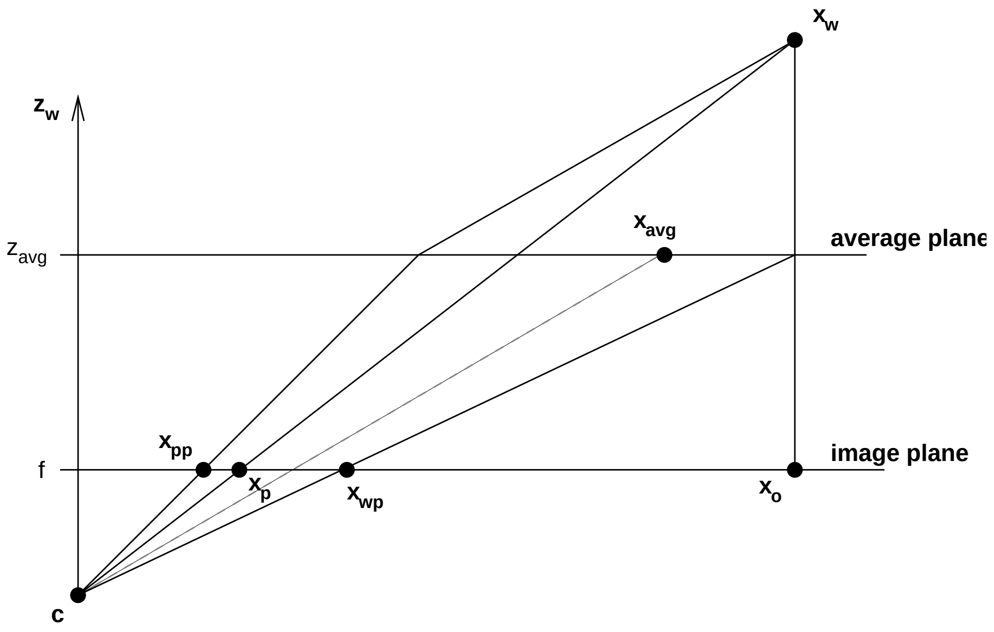


Figure 2.7: Action of different camera models. The images of the world point $\mathbf{x_w}$ are shown in the perspective ($\mathbf{x_p}$), the orthographic ($\mathbf{x_o}$), the weak-perspective ($\mathbf{x_{wp}}$), and the para-perspective model ($\mathbf{x_{pp}}$). Note that the camera center $\mathbf{c}$ corresponds only to the perspective model. The actual camera center of the other models lies at infinity. The figure just illustrates the action of the different models, not their actual way of projection.

for a displacement of the camera towards or away from the 3D scene. The weak-perspective projection is represented by the matrix:

$$\mathbf{P_{wp}} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 0 & z_{\text{avg}} \end{bmatrix} \tag{2.29}$$

Para-perspective projection For a large field of view, the fact that the points are first projected orthographically in the weak-perspective projection creates a large approximation error. In the para-perspective projection the points are first projected parallel along the direction defined by the camera center $\mathbf{c}$ and the average 3D scene point $\mathbf{x}_{\text{avg}} = (x_{\text{avg}}, y_{\text{avg}}, z_{\text{avg}})^T$ (dashed line in figure 2.7). The para-perspective projection is represented by the matrix:

$$\mathbf{P_{pp}} = \begin{bmatrix} f & 0 & -x_{\text{avg}}/z_{\text{avg}} & x_{\text{avg}} \\ 0 & f & -y_{\text{avg}}/z_{\text{avg}} & y_{\text{avg}} \\ 0 & 0 & 0 & z_{\text{avg}} \end{bmatrix} \tag{2.30}$$

This type of projection is a first order Taylor approximation to the perspective projection [Poelman & Kanade 97].

Affine cameras and traffic scenes Affine cameras are good approximations to the finite projective camera if the depth variation of the 3D scene is small compared to the average depth of the 3D scene. In traffic scenes this is not the case. Figure 2.8 shows a typical image. There are close objects as well as far objects. The size of the imaged objects is very different. Also, parallel world lines are not mapped to parallel lines in the image at all. Thus affine cameras are not applicable in traffic scenes. The full perspective camera must be employed.

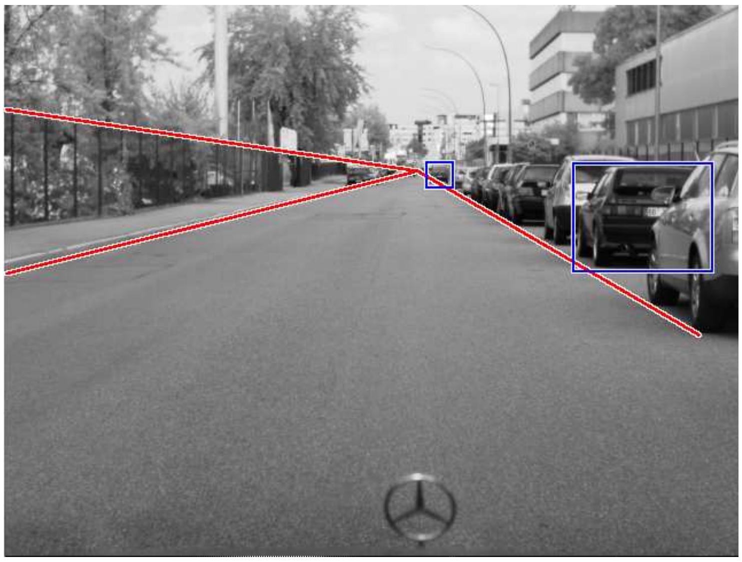


Figure 2.8: In traffic scenes the depth variation is very high. Parallel world lines made up by the curb, the fence, and the parking cars are not parallel in the image (red lines). Cars at different distances are imaged differently in size (blue rectangles).

2.3 Two View Geometry

2.3.1 Epipolar Constraint

If the 3D scene is seen by two cameras having a different viewpoint the images are related to each other. The images of one and the same world point satisfy a geometric constraint, called *epipolar constraint*. Figure 2.9 illustrates this constraint. The world point $\mathbf{x_w}$ is projected onto the first image plane yielding $\mathbf{x_1}$ and onto the second image plane yielding $\mathbf{x_2}$. We say that $\mathbf{x_1} \leftrightarrow \mathbf{x_2}$ are corresponding points, or shortly a *correspondence* .

Now imagine that $\mathbf{x_w}$ shifts along the viewing ray $\overline{\mathbf{c_1 x_1}}$ to the point $\mathbf{x'_w}$. The image of that point in the first view is still $\mathbf{x_1}$, whereas the image in the second view has changed to $\mathbf{x'_2}$. In particular, the image point has moved along the line $\overline{\mathbf{e_2 x_2}}$. This line arises from the intersection of two planes: the second image plane and the plane defined by the points $\mathbf{c_1}$, $\mathbf{c_2}$, and $\mathbf{x_1}$. The latter one is called the *epipolar plane* . The resulting intersection line is called the *epipolar line* . Also the points $\mathbf{e_1}$ and $\mathbf{e_2}$ have a special name: *epipole* . An epipole is the image in one view of

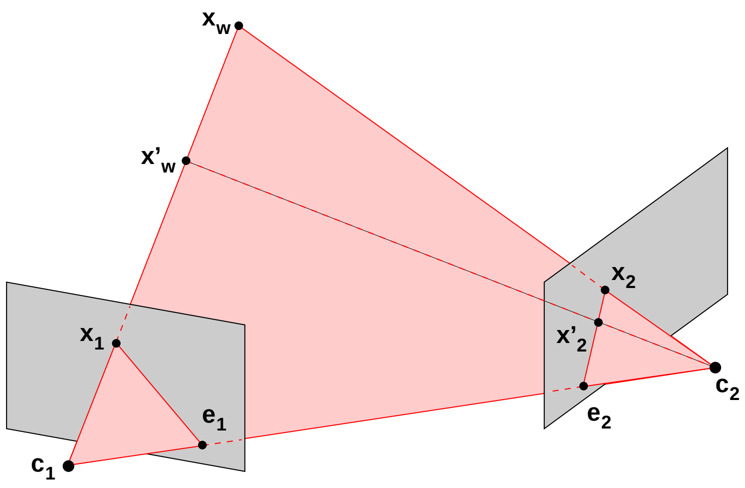


Figure 2.9: Epipolar constraint. A world point $\mathbf{x_w}$ moving along the viewing ray $\overline{\mathbf{c_1x_1}}$ is imaged as a line $\overline{\mathbf{e_2x_1}}$ in the second view.

the camera center of the other view or in other words it is the intersection of the line joining the camera centers (the *baseline*) with the image plane.

Epipolar constraint *Two image points* $\mathbf{x_1}$ *and* $\mathbf{x_2}$ *satisfy the epipolar constraint if and only if* $\mathbf{x_2}$ *lies on the epipolar line corresponding to* $\mathbf{x_1}$. *Alternatively, one may say that* $\mathbf{x_1}$ *has to lie on the epipolar line corresponding to* $\mathbf{x_2}$.

After this geometric excursion the algebraic representation of the epipolar geometry is discussed. Starting from the image point $\mathbf{x_1}$ we search for a world point lying on the viewing ray $\overline{\mathbf{c_1x_1}}$. The pseudo-inverse of the projection matrix $\mathbf{P_1}$ provides such a point. Lets recycle the term $\mathbf{x_w}$ for that point: $\mathbf{x_w} = \mathbf{P_1}^+\mathbf{x_1}$. The corresponding point in the second view is just: $\mathbf{x_2} = \mathbf{P_2 x_w}$. The epipolar line joining $\mathbf{e_2}$ and $\mathbf{x_2}$ is: $\mathbf{l_2} = \mathbf{e_2} \times \mathbf{x_2} = [\mathbf{e_2}]_\times \mathbf{x_2}$. Combining all three steps yields a 3×3 matrix, called *fundamental matrix*:

$$\mathbf{F} = [\mathbf{e_2}]_\times \mathbf{P_2 P_1}^+ \tag{2.31}$$

The fundamental matrix transforms points in the first view to corresponding epipolar lines in the second view: $\mathbf{l_2} = \mathbf{Fx_1}$. The above approach could have also started from the image point $\mathbf{x_2}$ in the second view which would end with the corresponding epipolar line $\mathbf{l_1}$ in the first view. This would lead to the transposed fundamental matrix, i.e. $\mathbf{l_1} = \mathbf{F}^T\mathbf{x_2}$. The algebraic representation of the epipolar constraint reads:

$$\mathbf{x_2}^T\mathbf{Fx_1} = 0 \tag{2.32}$$

This constraint is linear in the entries of $\mathbf{F}$ and bilinear in the entries of the correspondence $\mathbf{x_1} \leftrightarrow \mathbf{x_2}$. That is why equation 2.32 is sometimes called *bilinear constraint*.

If the calibration matrices $\mathbf{K_1}$ and $\mathbf{K_2}$ are known the fundamental matrix can be expressed in terms of normalized image coordinates: $\mathbf{x}_1' = \mathbf{K}^{-1}\mathbf{x_1}$ and $\mathbf{x}_2' = \mathbf{K}^{-1}\mathbf{x_2}$:

$$\mathbf{x_2}^T\mathbf{Fx_1} = \mathbf{x_2K}^{-T}\mathbf{FK}^{-1}\mathbf{x_1} = \mathbf{x}_2'^T\mathbf{Ex}_1' = 0 \tag{2.33}$$

The matrix $\mathbf{E} = \mathbf{K}^{-T}\mathbf{F}\mathbf{K}^{-1}$ is called *essential matrix*. It covers the relative location of the two cameras. If the world coordinate frame is identical to the first camera, i.e. $\mathbf{P_1} = \mathbf{K_1}[\mathbf{I}|\mathbf{0}]$ then $\mathbf{P_2} = \mathbf{K_2}\mathbf{R}[\mathbf{I}| - \mathbf{t}]$, and $\mathbf{E}$ simply computes to:

$$\mathbf{E} = [-\mathbf{R}\mathbf{t}]_\times \mathbf{R} \tag{2.34}$$

2.3.2 Triangulation

Looking at figure 2.9 it can be seen that given the correspondence $\mathbf{x_1} \leftrightarrow \mathbf{x_2}$ the 3D point $\mathbf{x_w}$ is reconstructable through intersection (*triangulation*) of the two viewing rays $\overline{\mathbf{c_1 x_1}}$ and $\overline{\mathbf{c_2 x_2}}$. Prerequisite is the knowledge about the location of the cameras to each other. In particular it means that the projection matrices must be known.

The triangulation fails if the viewing rays are co-incident. This holds for the viewing rays defined by the epipoles. The reconstruction in this case is ambiguous. All 3D points along the ray $\overline{\mathbf{c_1 e_1}} = \overline{\mathbf{c_2 e_2}}$ induce the same correspondence $\mathbf{e_1} \leftrightarrow \mathbf{e_2}$. Correspondences near the epipoles have almost co-incident viewing rays which results in inaccurate (noisy) reconstructions.

There are different triangulation methods in the literature. A fast but statistically not optimal method is the *direct linear transform* (DLT). Another - statistically optimal - method is the *optimal polynomial method* which minimizes the reprojection error. For details refer to [Hartley & Zisserman 03].

The triangulation works if the 3D point $\mathbf{x_w}$ is static. A moving 3D point in general is not reconstructable, due to a manifold ambiguity. However, if we place a constraint on the shape of the trajectory of the moving point, for instance a straight line or a conic section, the 3D point becomes reconstructable except for some degenerate cases.

In the case of the straight line five images (or to be more precisely five rays towards the moving point) are required to get a unique solution for the reconstruction. The solution is the generator line of a linear line complex including the rays, i.e. the line intersecting all rays. Once this line is calculated the 3D position of the moving point is determined by triangulation between the line and the single rays. That is why this method is called *trajectory triangulation* [Avidan & Shashua 00]. Figure 2.10 illustrates this.

Degenerate situations occur when the moving 3D point and the camera center trace trajectories that live in the same ruled quadric surface. Such a surface is generated by two sets of disjoint lines. Each line from one set meets each line from the other set. Any intersection of the surface with a plane yields a curve of second order. Ruled quadric surfaces are the hyperboloid of one sheet, the cone, two planes, the line, and the point.

2.4 Parameter Estimation

Within this thesis we will estimate parameters of certain models based on measurements. These measurements are related algebraically to the model, so given a sufficient number of measurements the parameters of the model can be computed. For example the model of a 2D line: $y(x) = m \cdot x + b$ is characterized by the two parameters: m and b. With two measurements (2D

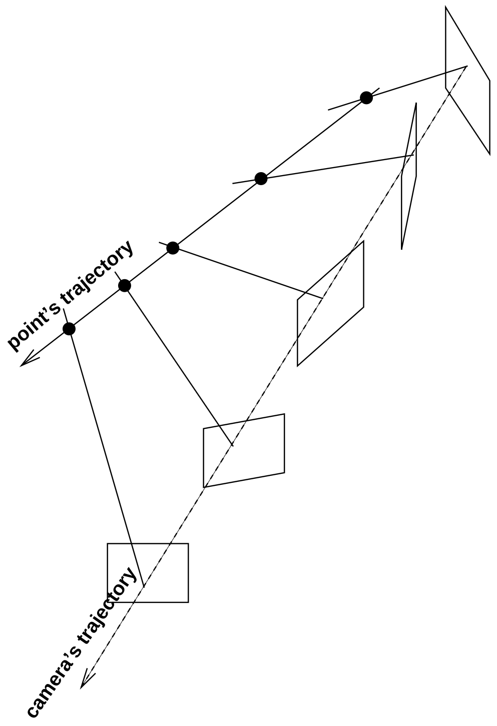


Figure 2.10: Trajectory triangulation. If the trajectory of a moving 3D point is a line five images of this point define it's trajectory uniquely.

points): $(\bar{x}_1,\bar{y}_1)$ and $(\bar{x}_2,\bar{y}_2)$ the parameters m and b are uniquely defined. In real life, however, the measurements are uncertain which prevents an exact computation of the parameters. They can only be estimated. In order to achieve accurate estimates one exploits the power of statistics: Increasing the number of measurements stabilizes the estimate.

Next section the least squares estimation method is discussed. Based on the 2D line example its effectiveness is shown. The example also shows that this method is vulnerable to gross errors in the measurements (*outliers*), i.e. the least squares estimate may be perturbed if outliers are present. For this reason methods were developed which are robust to outliers. Two of them are discussed in the sections 2.4.2 and 2.4.3. Section 2.4.4 compares the diferent methods where the 2D line serves as an example again.

2.4.1 Least Squares

The question now is how to get an estimate for m and b given a set of uncertain measurements $(\bar{x}_1,y_1),(\bar{x}_2,y_2),..,(\bar{x}_n,y_n)$? Note that only the y-component is subject to errors. The x-component is assumed error free (denoted by the bar accent). Of course, we want to get the best achievable estimate. To this end, we have to know the probability density function (PDF) of each individual measurement error (residual) $r_i = y(\bar{x}_i) - y_i$. The probability of observing a certain residual depends on $\bar{x}_i$, m, and b. For convenience we summarize m and b in the parameter vector $\mathbf{p} = (m,b)^T$. The probability then is given by $\text{pdf}_i(r_i|\bar{x_i},\mathbf{p})$. Assuming that the residuals are independent the joint probability of observing the entire set of residuals is:

$$L(\mathbf{p}) = \prod_i \text{pdf}_i(r_i|\bar{x_i},\mathbf{p}) \tag{2.35}$$

L is called the likelihood function. The parameter vector $\hat{\mathbf{p}}$ for which L becomes maximal:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} L(\mathbf{p}) \tag{2.36}$$

represent the best achievable estimate, since this parameter vector is the most likely one which has generated the given set of measurements (sample). The parameter vector achieved this way constitute a *maximum likelihood estimate* (MLE).

In practice one commonly assumes that the residuals obey a Gaussian distribution $r_i \sim N(0,\sigma)$. Then equation 2.36 becomes:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}} \prod_i e^{-\frac{r_i^2}{2\sigma^2}} \tag{2.37}$$

The normalization constant of the Gaussian distribution in equation 2.37 is omitted, since it does not effect the solution. Equation 2.37 is simplified by taking the negative logarithm:

$$\hat{\mathbf{p}} = \arg\min_{\mathbf{p}} \sum_i r_i^2 \tag{2.38}$$

yielding the well-known *least squares method*. Figure 2.11 demonstrates the effectiveness of this method, but also shows its limit. In figure 2.11a the 2D line is estimated based on two measurements only. Clearly, the estimate differs considerably from the true 2D line. In figure 2.11b the least squares method is applied using ten measurements. The estimated 2D line is very close to the true one. Figure 2.11c shows that gross errors in the measurements spoil the estimate. Such measurements are called *outliers*. The least squares method is not robust to outliers.

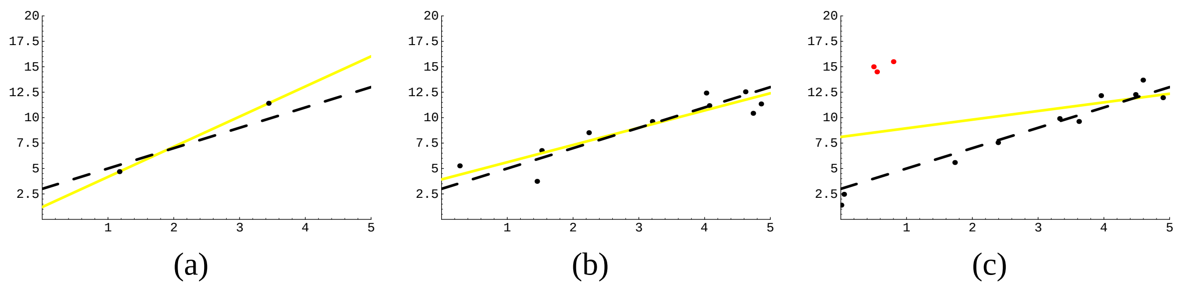


Figure 2.11: Least squares estimation. A 2D line (yellow) is estimated based on uncertain measurements (black dots). The true 2D line is marked by the dashed line. (a) Two measurements are required to compute the 2D line. The result is poor. (b) The least squares estimate based on ten measurements yields a good result. (c) Outliers (red dots) spoil the estimate.

The next sections deal with estimation methods which can handle outliers appropriately.

2.4.2 M-Estimation

We have seen that the least squares method constitute a maximum likelihood estimate if the residuals r_i are Gaussian distributed. Outliers, however, either are not Gaussian or have a higher variance than the inliers. The least squares method is not optimal in such cases.

To overcome this issue Huber proposed the generalized maximum likelihood estimation [Huber 81] and called it *M-estimation* where M stands for "maximum likelihood-type". His approach generalizes the square function in equation 2.38 to an arbitrary cost function $C = C(r)$. This allows to formulate MLE's for non-Gaussian distributed residuals. For example, if the inliers as well as the outliers are Gaussian distributed with standard deviations σ_{in} and σ_{out}, respectively, the PDF with the normalization constant omitted is $\text{pdf}(r) = \varepsilon \exp(-r^2/2\sigma_{in}^2) + (1-\varepsilon)\exp(-r^2/2\sigma_{out}^2)$ with ε the expected fraction of the inliers. The cost function then is:

$$C(r) = -\log\left(\varepsilon \exp(-r^2/2\sigma_{in}^2) + (1-\varepsilon)\exp(-r^2/2\sigma_{out}^2)\right) \tag{2.39}$$

The cost function 2.39 is called *corrupted Gaussian*. In contrary to the least squares function the corrupted Gaussian attenuates the influence of the outliers (fig. 2.12) which results in more robust estimates. In summary, if the distribution of the residuals is known, it is always possible to construct a MLE by setting $C(r)$ appropriately.

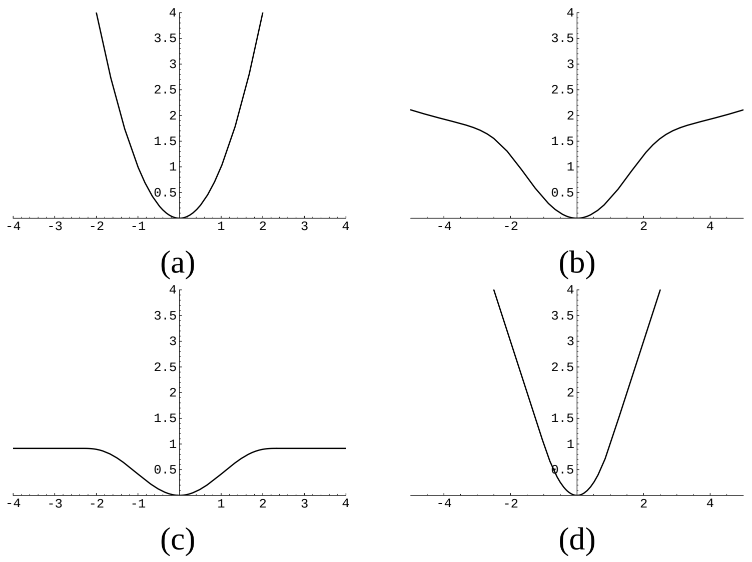


Figure 2.12: Different cost functions. (a) square function. (b) corrupted Gaussian with $\sigma_{\text{in}} = 1$, $\sigma_{\text{out}} = 5$, $\varepsilon = 0.8$. (c) Tukey with $\sigma_{\text{in}} = 0.5$. (d) Huber with $T = 1$

There are also cost functions motivated more by heuristics than by adherence to a specific noise-distribution model. A famous function is the *Tukey function* [Mosteller & Tukey 77]:

$$C(r) = \begin{cases} \frac{(c\sigma_{\text{in}})^2}{6}\left[1-\left(1-\left(\frac{r}{c\sigma_{\text{in}}}\right)^2\right)^3\right] & , |r| < c\sigma_{\text{in}} \\ (c\sigma_{\text{in}})^2/6 & , |r| \geq c\sigma_{\text{in}} \end{cases} \tag{2.40}$$

where $c = 4.6851$ is the tuning constant. The graph of this function is shown in figure 2.12c. The Tukey function is able to suppress the outliers completely, since $C(r)$ takes on a constant value for large residuals. The drawback of this function is its non-convexity. Thus, the sum of the Tukey evaluated residuals will have several local minima which can make convergence to the global minimum chancy. It should by applied only when an initialization near the global minimum is guaranteed.

The fourth and last cost function we discuss here, the *Huber function*, is convex and thus does not introduce additional local minima. The price we have to pay is a reduced robustness over the Tukey function. It is defined as:

$$C(r) = \begin{cases} r^2 & , |r| < T \\ 2T|r| - T^2 & , |r| \geq T \end{cases} \tag{2.41}$$

Residuals larger than T are treated as outliers. Their influence grows only linear instead of quadratic. The threshold T should be chosen to one to three times the inlier standard deviation. The graph of the Huber function is shown in figure 2.12d. We will use this function throughout the thesis.

Inherent in all robust cost functions is the knowledge about the inlier standard deviation σ_{in}. The robust estimation of it is related to the median of the absolute values of the residuals:

$$\hat{\sigma}_{\text{in}} = 1.4826\,[1 + 5/(N - \dim(\mathbf{p}))]\ \operatorname*{median}_i |r_i| \tag{2.42}$$

The magic number 1.4826 comes from the Gaussian normal distribution. The median of the absolute values of random numbers sampled from the distribution $N(0,1)$ is equal to $\Phi^{-1}(3/4) \approx 1.4826$. The term $1 + 5/(N - \dim(\mathbf{p}))$, with N the number of measurements, compensates for the effect of a small set of measurements. More about the theory of M-estimation can be found in [Maronna *et al.* 06].

2.4.3 RANSAC

RANSAC (RANdom SAmple Consensus)[1] proposed by [Fischler & Bolles 81] seeks to detect outliers by sampling and rating several minimal subsets from the given set of measurements. A minimal subset contains the minimal number of elements (measurements) required to compute the parameters of the model. In the case of the 2D line, two measurements (2D points) are required.

After a minimal subset was randomly sampled and the parameters were computed, the subset is rated based on the number of measurements consistent with the parameters. The higher the number the better the quality. This is done for a certain number of subsets. The best solution is the subset with the highest quality. Two points are not yet clarified: What does consistency mean and how many subsets should be sampled?

The original RANSAC method defines the consistency by means of the threshold function. A measurement is consistent if its residual r_i is smaller than the threshold. As in the case of

[1]By the way the website `www.ransac.org` has nothing to do with our RANSAC. It is the website of the Russian American Nuclear Security Advisory Council.

the robust cost functions the threshold should reflect the standard deviation of the inliers. Later works substitute the discrete threshold function by continuous ones.

For example MLESAC (Maximum Likelihood Estimation SAmple Consensus) proposed by [Torr & Zisserman 00] incorporate robust cost functions known from the M-estimation.

LMedS (Least Median of Squares) uses the very robust median [Rousseeuw 84]. Here the concept of consistent measurements is not appropriate but, nevertheless, the median states a good function providing the quality of a subset. The best solution is the subset with the lowest median of the squared residuals.

There is still the question how many subsets should be sampled? Since the measurements are contaminated by outliers, one subset is definetely insufficient. The hope is to collect a subset containing only inliers. Such a subset will provide a good estimate. The more subsets that are sampled, the higher the probability that at least one subset contains only inliers. Let the desired probability be P and the inlier fraction be ε, then the number of subsets M should be:

$$M \geq \frac{\log(1-P)}{\log\left(1-\varepsilon^{\dim(\mathbf{p})}\right)} \tag{2.43}$$

Since M may be large (> 50) RANSAC is computationally expensive. In recent years RANSAC has been accelerated. GASAC (Genetic Algorithm SAmple Consensus) [Rodehorst & Hellwich 06] for example samples subsets which are close to the best solution found so far. Preemptive RANSAC [Nistér 03] scores all subsets in parallel by testing the measurements successively. During this process bad subsets, having a low support, are rejected early which speeds up the computation.

Another problem with RANSAC is that the sampled measurements of a subset may lie close to each other making the estimate instable. Such subsets are useless and should be avoided. A method addressing this problem is GOODSAC (GOOd SAmple Consensus) [Michaelsen *et al.* 06]. It ensures that the measurements contained in a subset are uniformly distributed.

2.4.4 Comparison

We take up the 2D line example to show the robustness of the previously discussed estimation methods. Figure 2.13 shows the estimated 2D lines for different outlier fractions. With 23% outliers only the non-robust least squares method performs badly. All other methods (M-estimation with Huber function, RANSAC, LMedS) provide good estimates. When the outlier fraction is increased to 38% M-estimation as well as RANSAC reach their limit. The very robust LMedS still provides a good estimate. 54% outliers are an overkill. Note that these outlier fractions are just examples. They do not reflect the actual breakdown points of the individual methods.

The breakdown point of an estimation method is the smallest outlier fraction that can cause the estimator to take values arbitrary far away from the correct estimate. For least squares it is $1/N$ with N the number of measurements. M-estimation breaks down at $1-(1/2)^{1/\dim(\mathbf{p})}$ whereas LMedS takes on the maximum value of 50% independently from any parameters. [Stewart 99] compares the presented methods in more detail.

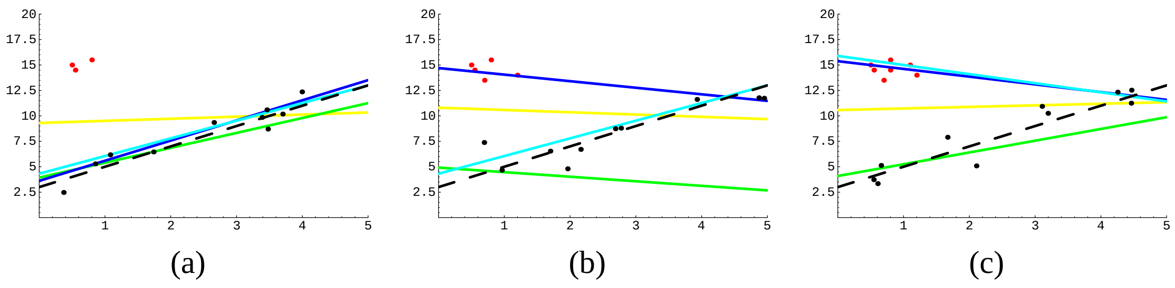


Figure 2.13: Robust estimation of a 2D line with (a) 23% outliers (b) 38% outliers and (c) 54% outliers. The true 2D line is marked by the dashed line. The least squares method (yellow) is not robust. M-estimation (green) and RANSAC (blue) get off at 38% outliers whereas LMedS (cyan) still performs well. No method is able to handle 54% outliers.)

Chapter 3

Optical Flow

The optical flow is the source of information on which the algorithms developed in this thesis rely on and thus deserves an extra chapter.

The optical flow represents local grey-value displacements from frame to frame. These displacements have two reasons: first because the camera and / or objects move through the scene and second because the illumination changes.

Illumination changes are manifold. The light source may change their spectrum of emitted light. Surfaces may vary the fraction of reflectance when the surface normal is rotated (diffuse and specular reflection). Structured light varying temporally causes moving shadows on illuminated objects. All three types occur in traffic scenes. When driving into and out of a tunnel the light source changes from the sun to a manmade lamp an back to the sun. In most cases the spectrum of the lamp is different from that of the sun. When cars are driving curves their surface normals rotate. Structured light is caused by the shadows of trees for example. To model all three types of illumination changes is cumbersome for natural scenes. The parameter space of a complete model is very high. The parameter estimation of such a model based on acquired images is infeasable due to lots of ambiguities. In practise, often simple illumination models are used, for example the linear model (scale + offset).

In contrast to image displacements induced by illumination changes the image displacements induced by a moving camera have exactly one reason, the motion. The parameter space of motion models is low. Hence, the estimation of the model parameters is feasable. Indeed, we will estimate them when we will deal with the ego-motion (chapter 4). In the next section two motion models are discussed. The entire set of image displacements computed by a motion model is called *motion field*.

In the second section we estimate the image displacements given two consecutive frames. The issues coming with the estimation are discussed as well as the estimation algorithm used in this thesis.

3.1 Motion Field

3.1.1 Discrete Motion Field

The discrete motion field describes the image displacements of projected 3D points caused by a moving camera. The projected motion of a 3D point between two frames is computed utilizing the projection matrices of the last camera $\mathbf{P_l}$ and the current camera $\mathbf{P_c}$. The latter one depends on the time interval Δt between the two frames. The choice of the world coordinate frame does not matter since it does not affect the image positions of the projected 3D points. For simplicity the world coordinate frame is set such that it coincides with the last camera. The last projection matrix then just contains the calibration matrix $\mathbf{K}$: $\mathbf{P_l} = \mathbf{K}\,[\mathbf{I}|\mathbf{0}]$. The current projection matrix arises from the transformation of the world coordinate frame into the current camera coordinate frame plus the projection onto the image plane: $\mathbf{P_c} = \mathbf{KR}\,[\mathbf{I}| -\mathbf{t}]$ (see also section 2.2.1).

The projected motion of a 3D point $\mathbf{x_w}$ then computes to:

$$\Delta\mathbf{x} = \tilde{\mathbf{x}}_\mathbf{c} - \tilde{\mathbf{x}}_\mathbf{l} \quad \text{with} \quad \mathbf{P_c x_w} \rightarrow \tilde{\mathbf{x}}_\mathbf{c}\,, \mathbf{P_l x_w} \rightarrow \tilde{\mathbf{x}}_\mathbf{l} \tag{3.1}$$

The tilde accent denotes inhomogeneous vectors.

The drawback of this discrete motion are the complex dependencies on the motion parameters. To see this we consider a simple example where the camera just rotates about the y-axis, i.e. $\mathbf{R} = \mathbf{R}(0, \Delta\psi, 0)$ and $\mathbf{t} = \mathbf{0}$. In order to get a simpler formula for $\Delta\mathbf{x}$ we normalize the image coordinates by applying $\mathbf{K}^{-1}$ to the image points. By doing this $\mathbf{K}$ becomes the identity matrix. The projected motion in normalized coordinates then reads:

$$\Delta\mathbf{x} = \frac{1}{\cos\Delta\psi + (\mathbf{x_l})_1 \sin\Delta\psi} \cdot \begin{pmatrix} -\left((\mathbf{x_l})_1^2 + 1\right)\sin\Delta\psi \\ -(\mathbf{x_l})_2\,(\cos\Delta\psi + (\mathbf{x_l})_1 \sin\Delta\psi - 1) \end{pmatrix} \tag{3.2}$$

This example shows that the discrete motion is non-linear in the motion parameter $\Delta\psi$, and further trigonometric functions are involved. However, equation 3.2 holds for arbitrarily large $\Delta\psi$'s. If the camera motion is small due to a small time interval Δt the motion field can be computed much simpler, which is described next.

3.1.2 Instantaneous Motion Field

The discrete motion field describes image displacements caused by an arbitrarily large time interval. In contrary the instantaneous[1] counterpart is only valid for infinitesimal time intervals. In practice infinitesimal time intervals are not possible but, nevertheless, the instantaneous motion field is a good approximation if the time interval is small.

The instantaneous motion field arises from differentiation of $\Delta\mathbf{x}$ (equation 3.1) with respect to Δt and setting $\Delta t = 0$:

$$\dot{\mathbf{x}} = \left.\frac{\partial\Delta\mathbf{x}}{\partial\Delta t}\right|_{\Delta t=0} \tag{3.3}$$

[1]In the literature the terms continuous and differential motion field are also found meaning one and the same.

With normalized image coordinates and given that $\mathbf{R}(\Delta t = 0) = \mathbf{I}$ and $\mathbf{t}(\Delta t = 0) = \mathbf{0}$ we get:

$$\dot{\mathbf{x}} = \frac{1}{z}\mathbf{A} \cdot \dot{\mathbf{t}} + \mathbf{B} \cdot \dot{\omega} \tag{3.4}$$

with

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -(\mathbf{x_I})_1 \\ 0 & 1 & -(\mathbf{x_I})_2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} -(\mathbf{x_I})_1(\mathbf{x_I})_2 & 1+(\mathbf{x_I})_1^2 & -(\mathbf{x_I})_2 \\ -(1+(\mathbf{x_I})_2^2) & (\mathbf{x_I})_1(\mathbf{x_I})_2 & (\mathbf{x_I})_1 \end{bmatrix} \tag{3.5}$$

and $z = (\mathbf{x_w})_3$ the depth of the 3D point. Since **A** and **B** only contain image coordinates equation 3.4 is linear in the motion parameters $\dot{\mathbf{t}} = \frac{\partial \mathbf{t}}{\partial \Delta t}\Big|_{\Delta t=0}$ and $\dot{\omega} = \frac{\partial \omega}{\partial \Delta t}\Big|_{\Delta t=0}$, where ω covers the three rotation angles. Also, the trigonometric functions have vanished making the computation of $\dot{\mathbf{x}}$ much simpler compared to that of $\Delta\mathbf{x}$. Note that the instantaneous motion field is characterized by the *translational velocity* $\dot{\mathbf{t}}$ and the *rotational velocity* $\dot{\omega}$ which is different from the discrete case. The instantaneous image displacement $\dot{\mathbf{x}}$ is called *image velocity*. It is the projection of the 3D velocity.

Figure 3.1 shows some exemplary motion fields caused by different camera motions and compares the discrete and the instantaneous motion field to each other. In the figure Δt is set to a high value $(\Delta t = 1)$ to point out the difference between the discrete and the instantaneous motion field. In case of a translation along the optical axis (fig. 3.1a) the instantaneous motion vectors are too short. In case of a horizontal translation (fig. 3.1b) the instantaneous motion field is equivalent to the discrete one. Figure 3.1c and 3.1d show rotations about the optical axis and the vertical axis, respectively. Here, the directions of the motion vectors change continuously over time causing an error in the instantaneous motion vectors.

3.1.3 Focus of Expansion

The motion field induced by a camera moving along the optical axis (fig. 3.1a) has a form like a star. The motion vectors seem to have a common origin. All extensions of the motion vectors intersect in the origin, the *focus of expansion* (FoE). When the camera moves backwards the origin is called focus of contraction.

The star-like motion field is preserved as long as the camera undergoes a pure translation. The FoE, in this case, points in the direction of travel, meaning that the viewing ray defined by the FoE and the camera translation **t** are parallel. In turn, it means that the FoE and the epipole in the last frame coincide.

If the camera rotates in addition, the focus of expansion does not exist. The motion vectors do not intersect in a common point, see figure 3.2. Note that there may be still points having zero motion, called *fixed points*. The set of 3D points inducing fixed points in the image is called *horopter*. In general the horopter is a twisted cubic (a curve of degree three in $\mathbb{P}^3$). [Verri *et al.* 89] characterizes fixed points as center, spiral, focus, node, saddle, or improper node. It also shows how the motion field looks like in the vicinity of such points. In figure 3.2 the fixed point is a spiral.

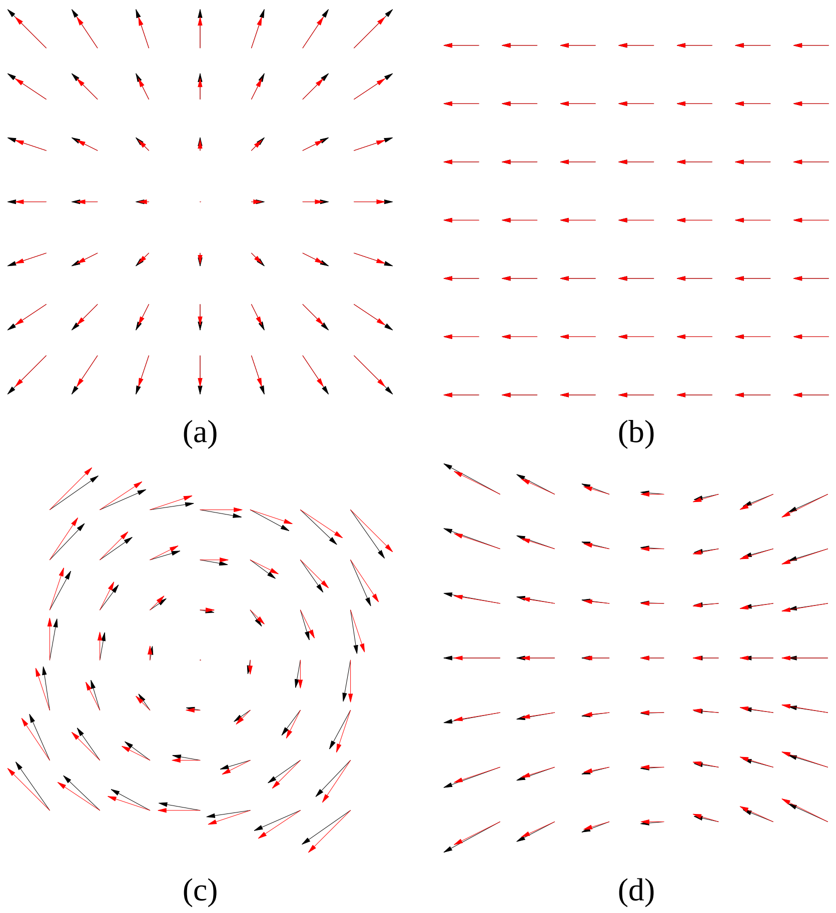


Figure 3.1: Discrete (black) vs. instantaneous (red) motion field. (a) Translation along the z-axis. (b) Translation along the x-axis. (c) Rotation about the z-axis with 10°. (d) Rotation about the y-axis with 10°.

3.2 Optical Flow

Last section we have seen how the motion field is computed caused by a moving camera. In practice, the motion parameters of the camera as well as the 3D structure of the scene are unknown. Thus, the motion field cannot be computed. Instead, it has to be determined directly from the images. In particular, the task is to find corresponding point pairs based on the similarity of local grey-value structures. This is not easy to accomplish since there are several hurdles to take:

- **Illumination change.** Physical illumination changes from frame to frame occur when the light source changes its output, or when diffuse or specular reflections change due to a rotation of 3D surfaces. Illumination changes also encounter when the camera adapts its exposure settings. In these cases the grey-value structures do not just "flow" over the image, but change their brightness, too. This spoils the similarity between them, which makes it hard to find corresponding point pairs.

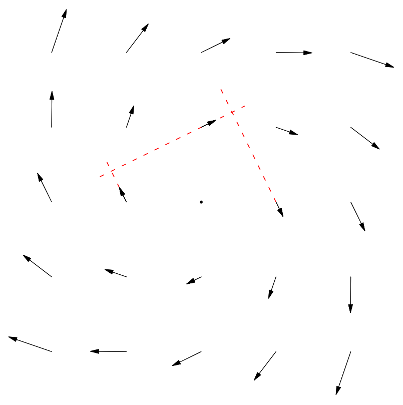


Figure 3.2: Non-existent focus of expansion. Motion field induced by a camera moving along and rotating about the optical axis. The dotted lines show that there is no common intersection point of the motion vectors.

- **Aperture problem.** If the local grey-value structure occurs multiple times in the image there are also multiple matching candidates. The correct one cannot be found. This problem arises especially at long grey-value edges induced by lane markings for example. Also low textured image regions, i.e. regions with low grey-value variations, suffer from this problem.

- **Occlusion.** A 3D point in the background seen in one frame is not seen in the other frame if the foreground occludes the background. Consequently, a corresponding point pair associated with that 3D point does not exist. The problem is that occlusions are not known a priori. An algorithm still tries to find matching grey-value structures and may give wrong results.

Figure 3.3 shows an example of this *correspondence problem*. The grey-value structure inside the green image patch is unique in the image. It is no problem to find the matching patch in the other image. In contrast to this, the red patch containing the curb occurs multiple times. There is no unique matching patch. The blue patch is just seen in one image. The oncoming vehicle occludes it in the other image.

Due to the problems mentioned above the apparent displacements of grey-value structures may be different from the actual displacements defined by the motion field. It means we determine the former one, which is called *optical flow field*, and hope that it is sufficiently close to the latter one.

Unfortunately, the term "optical flow" is not used consistently in the literature. Some authors, e.g. [Vidal 05], link it to the instantaneous motion field. They speak of optical flow if the displacements are infinitesimal or very small at least. If the discrete motion field is applied due to large displacements they speak of correspondences. But what does small and large mean? There is no strict threshold separating these terms.

In [Haussecker & Spies 99] the differentiation is made upon the way the displacements are estimated. Optical flow-based techniques "try to minimize an objective function pooling con-

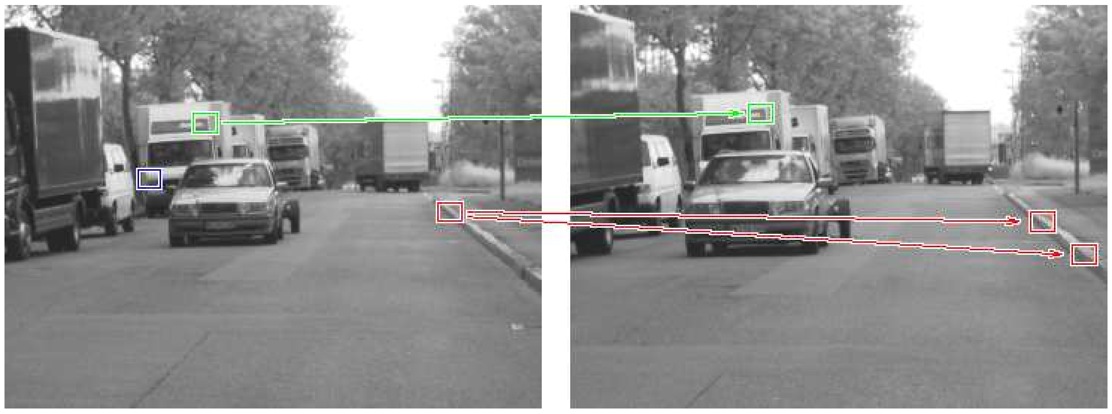


Figure 3.3: Correspondence problem. Two images taken at distinct time instances are shown. The green image patch matches uniquely. The red patch has multiple matches. The blue patch has no match.

straints over a small finite area". These techniques fail if the temporal sampling theorem is violated. "Correspondence-based techniques try to estimate a best match of features ...". "They are also capable of estimating long-range displacements ...".

In this thesis, the terms optical flow and correspondence have identical meanings. They both denote a corresponding point pair, regardless of the magnitude of the displacement between them.

The next section describes the optical flow algorithm used in this thesis. The literature explains plenty of other flow algorithms. They are not discussed further since the establishment of correspondences is beyond the scope of this thesis. The reader is referred to [Jähne 05, Haussecker & Spies 99] which give a good overview.

3.2.1 Census Transform based Estimation

The requirements to an optical flow algorithm depend heavily on the application. In the field of driver assistance the requirements are:

- real-time capability
- ability to handle large image displacements
- robustness to illumination changes

The flow algorithm developed by [Stein 04] meets these requirements. It uses the census transform as the representation of local image patches. The search for correspondences is done using a table based indexing scheme. In detail the method works as follows:

The census transform as applied in [Stein 04] compares the center pixel $\mathbf{x}$ of an image patch to the other pixels $\mathbf{x}'$ inside the patch:

$$\xi(I,\mathbf{x},\mathbf{x}') = \begin{cases} 0 & , \quad I(\mathbf{x}) - I(\mathbf{x}') > \varepsilon \\ 1 & , \quad |I(\mathbf{x}) - I(\mathbf{x}')| \leq \varepsilon \\ 2 & , \quad I(\mathbf{x}) - I(\mathbf{x}') < \varepsilon \end{cases} \tag{3.6}$$

with $I(\mathbf{x})$ the grey-value (intensity) at $\mathbf{x}$. The census digit ξ just measures the similarity between the grey-values at $\mathbf{x}$ and $\mathbf{x}'$. Typically, $\varepsilon = 12\dots16$. This representation is very robust to noise and is insensitive to a wide range of illumination changes.

All census digits of the image patch are clockwise unrolled building the *signature vector*. Figure 3.4 illustrates this. The signature vector is used to search for corresponding point pairs.

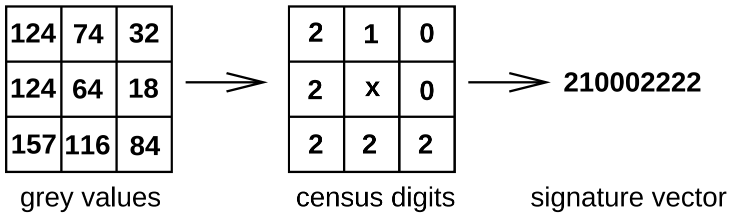


Figure 3.4: Census transform of 3×3 image patch.

To this end, all signature vectors of the first image are stored in a hash-table together with their pixel position. Then, all signature vectors of the second image are compared to the hash-table entries. This gives a list of putative correspondences (hypotheses) for each signature. The list is empty if a signature in the second image does not exist in the first image. In the event of multiple entries, the list is reduced by applying some photometric and geometric constraints. If there are still multiple entries, the one with the shortest displacement is taken. Thanks to the indexing scheme, arbitrary large displacements are allowed. Even when an image patch moves from the top left image corner to the buttom right corner it is matched.

The method is summarized in algorithm 3.1, with an example of its use shown in figure 3.5.

Comparison to Ground-Truth

The flow field retrieved by this algorithm is compared to ground-truth in order to measure its accuracy. An artifical scene rendered with OpenGL serves as a source for the ground-truth data. Since the 3D structure of the scene as well as the camera motion are known the motion field can be computed. Illumination changes are not present, so the optical flow field is identical to the motion field. Figure 3.6 shows an image of this artifical scene together with the measured and the ground-truth optical flow field. Some measured flow vectors are of unexpected large magnitude for instance at the street-lamp. They are obviously mismatched. The error histograms are shown in figure 3.7. The peaks at the margins collect all errors less than -2 pixels and greater than +2 pixels, respectively. Flow vectors having these errors are probably mismatched and treated as outliers. From the histograms we compute:

Algorithm 3.1 Optical Flow

1. Scan first image. Compute signature vector for each pixel $\mathbf{x}$:

$$\mathbf{s_1}(\mathbf{x}) = \bigotimes_{\mathbf{x}' \in D} \xi(I_1, \mathbf{x}, \mathbf{x}')$$

with $\bigotimes$ the concatanation operator and D the image patch centered at $\mathbf{x}$.

2. Filter out useless signatures. Patches containing no grey-value corners are vulnerable to the aperture problem. They are not processed further.

3. Store signature in hash-table. Signature vector $\mathbf{s}(\mathbf{x})$ is interpreted as a decimal number and serves as the key to the hash-table in which the center pixel $\mathbf{x}$ is stored.

4. Filter out useless signatures. If one and the same signature occurs too frequently it is deleted from the table. It is very likely that this signature occurs also frequently in the second image, so a unique correspondence will not be found.

5. Scan second image. Compute signature vector for each pixel $\mathbf{x}$:

$$\mathbf{s_2}(\mathbf{x}) = \bigotimes_{\mathbf{x}' \in D} \xi(I_2, \mathbf{x}, \mathbf{x}')$$

6. Compare signature vectors. Look for each $\mathbf{s_2}(\mathbf{x})$ in the hash-table whether there are one or more entries with the same signature vector.

7. Establish correspondence hypotheses. All point pairs $\mathbf{x_1}$, $\mathbf{x_2}$ with $\mathbf{s_1}(\mathbf{x_1}) = \mathbf{s_2}(\mathbf{x_2})$ are correspondence hypotheses.

8. Reduce the number of hypotheses. There may be several point pairs with identical signatures. Filter out the hypotheses where the illumination change is too high (e.g. $> 20\%$) or where the displacement $\|\mathbf{x_1} - \mathbf{x_2}\|$ is too high (e.g. > 70px). From the remaining hypotheses take the one with the shortest displacement.

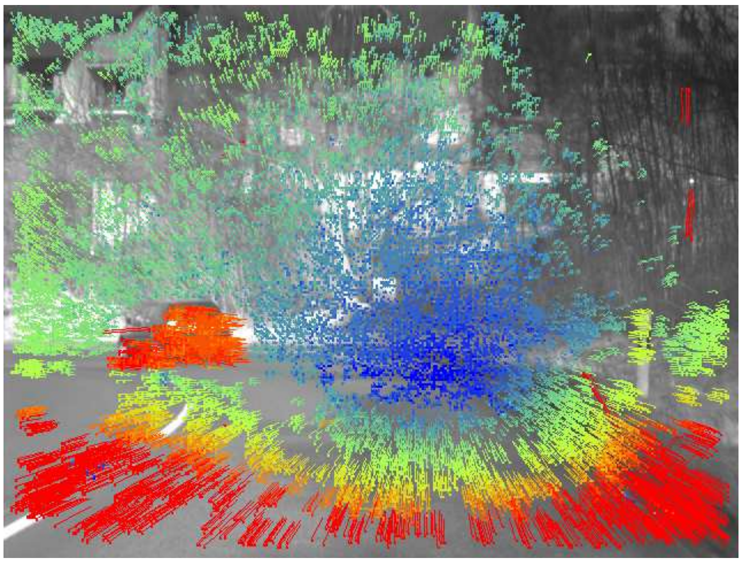


Figure 3.5: Optical flow field. The length of the flow vectors is color coded from blue (0px) to red (> 20px). There are 27639 vectors in total. The images were taken by a VGA camera with 12bit resolution.

	horizontal dir.	vertical dir.
mean	-0.0013px	-0.0004px
std. dev.	0.538px	0.483px
outliers	6.3%	4.8%

A standard deviation of about half a pixel is not surprising, because the flow algorithm is "only" pixel precise. Other flow algorithms achieve sub-pixel precision (typical accuracy 0.1px), however, they are computationally more expensive. An example is KLT, which stands for the inventors Kanade, Lucas, and Tomasi [Tomasi & Kanade 91, Shi & Tomasi 94]. The pixel precision property prevents the ability to track features, i.e. to establish correspondences (with high accuracy) over more than two frames.

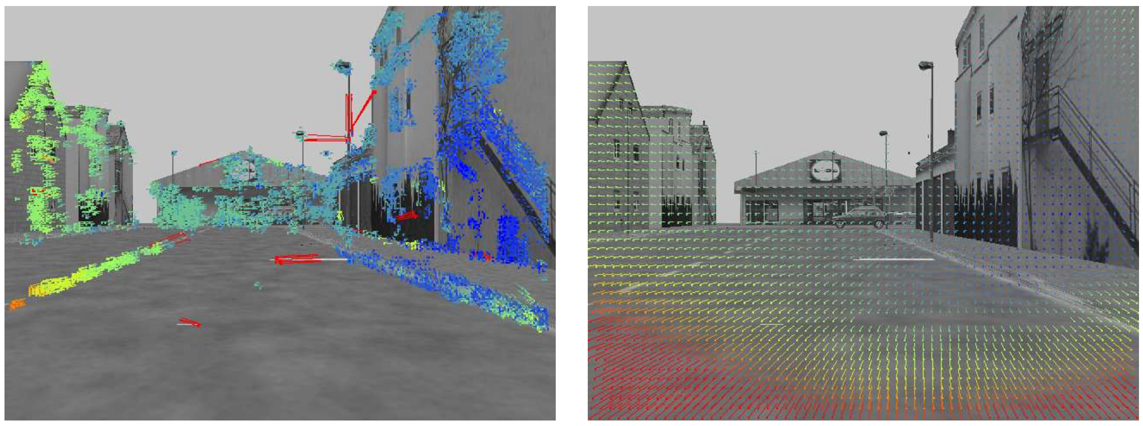


Figure 3.6: Comparison of the measured optical flow field to ground-truth data. (a) The flow algorithm applied to an artifical scene. (b) The ground-truth optical flow field.

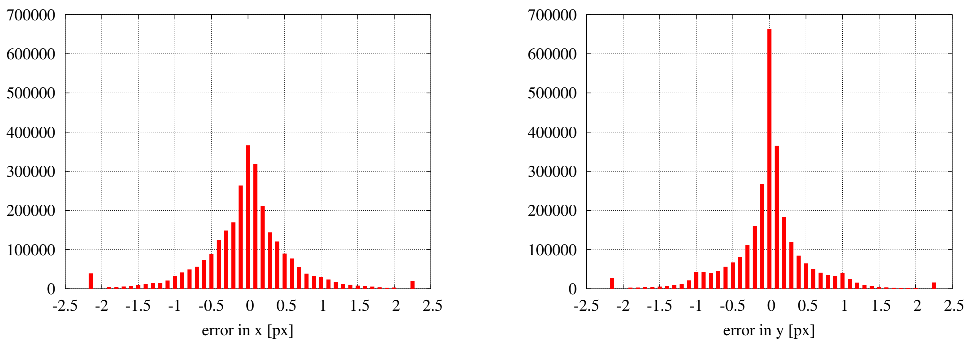


Figure 3.7: Error in the measured optical flow field. (a) Error histogram in horizontal direction. (b) Error histogram in vertical direction.

Chapter 4

Ego-Motion Estimation

A reconstruction of the 3D scene seen by two cameras is required in order to detect moving objects. The scene can be reconstructed only if the relative orientation of the two cameras to each other is known (section 2.3.2). In stereo vision, the cameras are rigidly mounted enabling the possibility to determine the relative orientation through offline calibration. However, in monocular vision, the camera relative orientation changes continueously due to the ego-motion. Consequently, the relative orientation (ego-motion) has to be determined in each frame.

This desirable information could be obtained from accurate *inertial measurement units* (IMU). Fully featured IMU's are equipped with 3 linear acceleration and 3 gyroscopic acceleration sensors. They measure all 6 degrees of freedom. In practice one is faced to two issues related to the use of IMU's. The first is that the IMU has to be coupled rigidly to the camera, otherwise the IMU will not reproduce the camera ego-motion adequately. The second is that the IMU and the camera have to be calibrated to each other. The literature addresses both issues. In [Chalimbaud *et al.* 05] a visuo-inertial sensor is presented which brings a CMOS imager (camera) close to a 6 DoF IMU. This compact design guarantees the rigidity. The calibration issue is addressed in the works [Lobo & Dias 05] and [Lang & Pinz 05].

Alternatively to IMU's the ego-motion may also be estimated utilizing the images directly. The disadvantage of vision is that the ego-motion estimation does not work well in all situations (e.g. at night, or during bad wheather where the optical flow field is sparse and noisy). The IMU's on the other side are expensive. An additional question is whether they provide accurate results within the entire velocity range. Another advantage of vision is that the estimated ego-motion is inherently synchronous to the acquired images. No timestamp battle!

In this chapter the vision based estimation of the ego-motion is considered. Next section the ego-motion is explained in detail. A comprehensive study on existing ego-motion estimation schemes follows. Based on it an appropriate scheme is selected (section 4.3) and explained (sections 4.4 and 4.5). In section 4.6 this scheme is extended by a motion model. It includes the iterative minimization of a non-linear function. Section 4.7 is dedicated to that issue. The advantages of the motion model are pointed out in section 4.8. The sensitivity analysis in section 4.9 shows that the image regions contribute differently to the estimate. The chapter ends with experimental results (section 4.10).

4.1 Ego-Motion in Detail

The camera undergoes an Euclidean transformation from frame to frame, consisting of three rotations and three translations along the coordinate axes. Given nothing else than the optical flow five out of these six degrees of freedom can be estimated. The length of the baseline (driven distance) between the two frames stays undetermined. The reason for this is found in the equation of the instantaneous motion field 3.4: One can simultaneously multiply the depth z and the translational velocity $\dot{\mathbf{t}}$ by an arbitrary scale factor λ without changing the image velocity $\dot{\mathbf{x}}$:

$$\dot{\mathbf{x}} = \frac{1}{z}\mathbf{A}\cdot\dot{\mathbf{t}} + \mathbf{B}\cdot\dot{\omega} = \frac{1}{\lambda z}\mathbf{A}\cdot\left(\lambda\dot{\mathbf{t}}\right) + \mathbf{B}\cdot\dot{\omega} \tag{4.1}$$

Fixing the scale factor requires the knowledge about either the depth z of at least one point or the magnitude of the velocity $\|\dot{\mathbf{t}}\|$ or the distance in 3D of at least two points (e.g. the height of a house). This *scale ambiguity* can be explained intuitively: Looking out of a locomotive while driving through the landscape one does not know whether the "universe" is real or a model railway. The observed scene as well as the motion are identical in both cases.

If image points are tracked over time and the initial driven distance (distance between the first two views) is known the distances between the upcoming views are determinable [vdHengel *et al.* 07].

4.2 Ego-Motion Estimation Schemes in the Literature

The problem of the reconstruction of the 3D scene seen by two cameras has attracted researchers for more than 100 years. The physicist and physiologist Herrmann von Helmholtz was the first who investigated the human ability to see three dimensional. He published his work "Handbuch der physiologischen Optik" in 1867. It was translated into english in 1925 [vHelmholtz 25]. Also the psychologist James J. Gibson [Gibson 50] has dealt with the visual perception of animals and humans. The term "optical flow" traces back to him.

Longuet-Higgins and Prazdny [LonguetHiggins & Prazdny 80] first published a method for estimating the full ego-motion, meaning translation and rotation. They show that the instantaneous optical flow (2D velocities) is composed by the sum of the rotational velocities and the translational velocities. The rotational velocities are smooth over the entire image and independent from the scene structure whereas the translational velocities are only smooth if the depth variations in the scene are continuous (compare to equation 3.4). This fact can be exploited to separate the translation from the rotation: Optical flow vectors in a local neighbourhood have an almost equal rotation part. Taking the difference of adjacent optical flow vectors cancels out the rotation. The difference between the translations, called motion parallax, remains. This vector points towards or away from the focus of expansion (FoE). If there is no depth discontinuity in the scene the motion parallax vector is zero, i.e. the translation part also cancels out, which is fatal. This is the drawback: A depth discontinuity is required but the measurement of optical flow at discontinuities is difficult.

During the years the literature has produced a wealth on ego-motion estimation schemes driven by the photogrammetry and robotics. Next, five properties are discussed on which the schemes are characterized:

- **Direct vs. Optical Flow:** Direct methods warp image patches according to the estimated ego-motion and compare the grey-values of the original patch in the last frame with the warped patch in the current frame. This avoids the computation of the optical flow. Low textured regions can also be taken into account. Direct methods need knowledge about the scene structure (e.g. the homography when looking at a scene plane) otherwise a warping would not be feasible. When there is no *a priori* knowledge the scene depth of every single point can be included as a parameter in the estimation process. However, this would increase the computional effort considerably (see [Mandelbaum *et al.* 98]). Sometimes direct methods are called *correspondenceless* methods.

- **Discrete vs. Instantaneous:** Instantaneous approaches employ the instantaneous motion field (section 3.1.2). They are applicable when the image displacements are small. The equations involved when using the instantaneous epipolar constraint or the instantaneous motion parallax are more tractable than the discrete counterparts (no trigonometric functions are required). There is no work known to the author which investigates the break down point of the instantaneous methods. So it is not clear what "small displacement" really means.

- **All Parameters together vs. Splitting of the Parameters:** Splitting the parameters (commonly into the translational and the rotational parameters) reduces first the search space and second resolves the ambiguity between translation and rotation [Tian *et al.* 96].

- **Two-View vs. Multiple View:** Two-view approaches just consider two consecutive images. They avoid tracking points over time. This is advantageous in siutations where tracking is infeasible. Rainy scenes with the windscreen wiper activated or scenes at night having a low image contrast are examples where tracking is problematic. Multiple view approaches are more powerful in the motion segmentation. Furthermore the multiple measurements lead to more accurate estimates. A good overview of the work on ego-motion estimation in "long" image sequences can be found in [Shariat & Price 90] and [Wu *et al.* 95].

- **Motion parameters only vs. Additional nuisance parameters:** Some approaches estimate not only the ego-motion parameters but also scene structure parameters (e.g. locations of individual 3D points), though we are not interested in such parameters. Their incorporation improves the estimates but also increases the computational complexity.

The literature on ego-motion estimation is so rich that one can combine the above properties almost arbitrarily and one will find at least one method with these properties. Hence it is quite hard to put the crowd of methods into a relational order making comparisons even more cumbersome. Indeed, there is a very limited number of papers comparing the different methods ([Tian *et al.* 96], [Armangué *et al.* 02], [Zhang & Tomasi 02]).

In the following sections some representative methods are discussed briefly.

4.2.1 Direct Methods

Grey-value domain. Direct methods define an error metric based on the grey-values which they minimize over the ego-motion and scene structure parameters. The sum of squared differences (SSD):

$$\text{ssd} = \sum_{\mathbf{x_l}} [I_l(\mathbf{x_l}) - I_c(f(\mathbf{x_l}))]^2 \tag{4.2}$$

is a common error metric. The grey-values (intensities) are denoted by I_l and I_c. The function f transforms a point in the last frame into the current frame. It depends on the ego-motion and scene structure parameters. In the most general case each point $\mathbf{x_l}$ has its own depth. All the depth values have to be estimated together with the ego-motion parameters which would be an overkill. The solution is to model the scene. A 3D plane, for example, has only three parameters (normal vector $\mathbf{n}$ plus distance d to origin). So, if the camera looks at a plane only these three scene structure parameters have to be estimated.

The methods [Stein *et al.* 00] and [Ke & Kanade 03] model the road as a plane and consider only the image region where the road is present. The SSD inside this region is minimized. The general idea behind this is illustrated in figure 4.1. The warping function f depending on

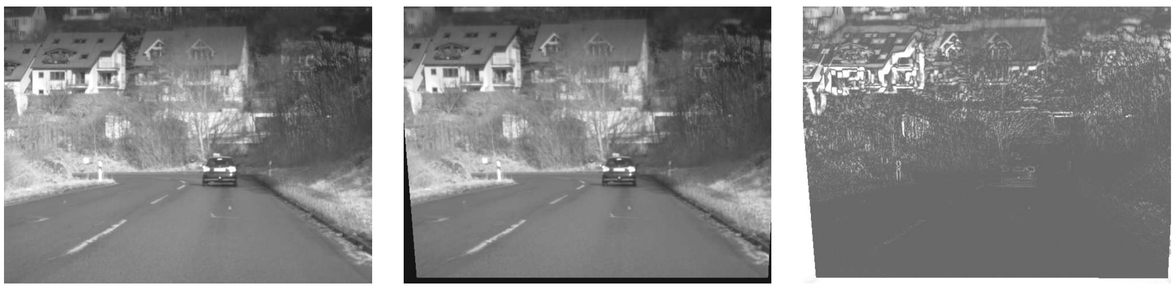

(a) (b) (c)

Figure 4.1: Direct method for estimating the ego-motion. The last image (a) is subtracted from the warped current image (b) yielding the difference image (c). The grey-value differences on the road are minimal.

the parameters: $\mathbf{t}, \omega, \mathbf{n}, d$ virtually transforms the current camera to a new location. If the new location is equal to the last camera's location the warped current image is the same as the last image. Hence, the grey-value difference (fig. 4.1c) is zero. One says that the image regions (containing the road plane) are registrated.

Frequency domain. It is also possible to estimate the ego-motion in the frequency domain. Strictly speaking methods doing so are neither direct methods nor optical flow methods. Nevertheless, they are treated as direct methods because they share the idea of using the complete information included in the images. Frequency based methods manage cluttered 3D scenes (a scene containing, for example, bushes and trees). Such scenes are the natural enemy of optical flow algorithms due to the high number of occlusions.

One representative is [Langer & Mann 04] which estimates the translational direction of the camera. It is assumed that there is no rotation. In local image patches the image velocities (instantaneous motion field) all lie on a line. The position within the line depends on the image position and scene depth. The spatio-temporal image cube is transformed into the 3D frequency domain. In that domain the motions lie on planes all intersecting in a common line, depending on the translational direction. The proposed algorithm searches for that line. In [Mann & Langer 05] the method was extended to motions containing rotations.

In [Makadia *et al.* 05] the Radon transform is employed to estimate the ego-motion. A correlation integral is formulated measuring how well the epipolar constraint is met given particular ego-motion parameters (**R** and **t**):

$$G(\mathbf{R},\mathbf{t}) = \int_{\mathbf{x_l}} \int_{\mathbf{x_c}} g(\mathbf{x_l},\mathbf{x_c}) \cdot \Delta(\mathbf{R}\mathbf{x_l},\mathbf{x_c},\mathbf{t})\, d\mathbf{x_l}\, d\mathbf{x_c}$$

$g(\mathbf{x_l},\mathbf{x_c})$ measures the similarity between the two image positions $\mathbf{x_l}$ and $\mathbf{x_c}$. The authors uses the Euclidean distance of SIFT[1] features computed at the positions $\mathbf{x_l}$ and $\mathbf{x_c}$. The Δ function measures how close the image positions $\mathbf{x_l}$ and $\mathbf{x_c}$ comes to satisfying the epipolar constraint. The maximum value of $G(\mathbf{R},\mathbf{t})$ gives the ego-motion which is searched for. However sampling $G(\mathbf{R},\mathbf{t})$ would result in a combinatorial explosion. The authors avoid this sampling by the application of the spherical Fourier transform. The computation of $G(\mathbf{R},\mathbf{t})$ in the Fourier domain is much easier. The approach is very robust against outliers. However, the computational burden prevents the algorithm of being real-time capable.

The method developed by Domke and Aloimonos [Domke & Aloimonos 06] is a hybrid methd. It computes correspondence candidates based on the phase of tuned Gabor filters. The more similar the responses between two image patches are the higher the correspondence probability is. The ego-motion then is estimated maximizing the joint probability. Repetitive patterns in the image inducing ambiguities in the optical flow are managed by this method.

4.2.2 Optical Flow Methods

Instantaneous motion. Ego-motion estimation methods based on the instantaneous motion field as described in section 3.1.2 typically minimize some sort of:

$$\sum_{i=1}^{N} ||\dot{\mathbf{x}}_{\mathbf{m},i} - \frac{1}{z_i}\mathbf{A}_i \cdot \dot{\mathbf{t}} - \mathbf{B}_i \cdot \dot{\omega}||^2 \tag{4.3}$$

with $\dot{\mathbf{x}}_{\mathbf{m},i}$ the i-th measured image velocity. This error metric requires to minimize over the depth values z_i (nuisance parameters), too. Methods have been developed reducing these parameters or even eliminating them completely. [Bruss & Horn 81] imposes a constraint on the depth:

$$z = \frac{||\mathbf{A}\dot{\mathbf{t}}||^2}{(\dot{\mathbf{x}}_{\mathbf{m}} - \mathbf{B}\dot{\omega})^T \mathbf{A}\dot{\mathbf{t}}} \tag{4.4}$$

[1]Scale Invariant Feature Transform [Lowe 99]

This effectively chooses the depth minimizing the distance of $\dot{\mathbf{x}}_\mathbf{m}$ to the instantaneous epipolar line. For computational ease the authors drop the term $||\mathbf{A}\dot{\mathbf{t}}||^2$ in equation 4.4. Putting this into equation 4.3 yields:

$$\sum_{i=1}^{N} |\,(\dot{\mathbf{x}}_{\mathbf{m},i} - \mathbf{B}_i\dot{\omega})_1 \left(\mathbf{A}_i\dot{\mathbf{t}}\right)_2 - (\dot{\mathbf{x}}_{\mathbf{m},i} - \mathbf{B}_i\dot{\omega})_2 \left(\mathbf{A}_i\dot{\mathbf{t}}\right)_1 |^2 \tag{4.5}$$

This error metric imposes a *bilinear constraint* on the ego-motion parameters which is equivalent to the *instantaneous epipolar constraint*. It is minimized easily, however, the estimate is biased. The reason is the improper scaling of the residuals caused by dropping of $||\mathbf{A}\dot{\mathbf{t}}||^2$. In the end an algebraic error is minimized instead of a geometric error. The RM-L1.2 method [Zhang & Tomasi 02] takes $||\mathbf{A}\dot{\mathbf{t}}||^2$ into account leading to unbiased and consistent estimtates. For robustness the square function in expression 4.5 is substituted by $|\cdot|^{1.2}$:

$$\sum_{i=1}^{N} \left| \frac{(\dot{\mathbf{x}}_{\mathbf{m},i} - \mathbf{B}_i\dot{\omega})_1 \left(\mathbf{A}_i\dot{\mathbf{t}}\right)_2 - (\dot{\mathbf{x}}_{\mathbf{m},i} - \mathbf{B}_i\dot{\omega})_2 \left(\mathbf{A}_i\dot{\mathbf{t}}\right)_1}{||\mathbf{A}\dot{\mathbf{t}}||} \right|^{1.2} \tag{4.6}$$

The error metric 4.6 is appropriate if the noise in the measured optical flow is homoscedastic, i.e. independent from the length $||\dot{\mathbf{x}}||$. For heteroscedastic noise different scalings are required. The appropriate scaling for noise proportional to the translational component of the optical flow is presented in [Zhu *et al.* 05].

In contrast to [Bruss & Horn 81], which completely eliminates the depth values in expression 4.3, [Zucchelli *et al.* 02] reduces the number of depth values by incorporating 3D lines and planes. For all points on a plane with the normal vector $\mathbf{n}$ and the distance d it holds:

$$z = \frac{d}{\mathbf{x_l}^T \mathbf{n}} \tag{4.7}$$

There are only three scene structure parameters to estimate ($\mathbf{n}$, d) in addition to the ego-motion parameters.

[Jepson & Heeger 90] split the ego-motion parameters by an algebraic manipulation of the instantaneous motion field. The resulting constraints depend only on the translation $\dot{\mathbf{t}}$. This allows the estimation of $\dot{\mathbf{t}}$ seperately from $\dot{\omega}$. Their method, which they called the *subspace method*, should attract a lot of researchers later on. The authors themselves developed a version which is linear on $\dot{\mathbf{t}}$ and found out that the outcoming estimates are biased [Heeger & Jepson 92].

[Lawn & Cipolla 96] introduced the linearised subspace method, which is applicable if small image patches are considered. Here, only four image points are needed to extract a constraint on $\dot{\mathbf{t}}$ instead of seven as needed in [Heeger & Jepson 92]. This has advantages for outlier rejection and may also improve the stability of the solution with respect to noise on the optical flow measurements.

The multiple view method in [Soatto & Perona 97] takes the subspace method as a basis to formulate a recursive estimation of the ego-motion using two Kalman filters.

Another method [Irani *et al.* 97] splitting $\dot{\mathbf{t}}$ and $\dot{\omega}$ first searches for an image region looking at a planar surface and then estimates the instantaneous homography registrating the last and the current region. This cancels out the rotational velocity $\dot{\omega}$. The translational velocity $\dot{\mathbf{t}}$ is estimated from the residual motion parallax.

[Pauwels & Hulle 04] addresses the issue of *image stabilization,* which is closely related to ego-motion estimation. The objective here is to get rid of the vibrations in the images caused by camera rotations. To this end, the rotational component of the ego-motion has to be estimated. [Pauwels & Hulle 04] employs a phase based optical flow algorithm and minimizes the vibrations of the optical flow in the phase realm.

[Baumela *et al.* 00] uses the instantaneous epipolar constraint to estimate the ego-motion. The results, when compared to the discrete counterpart, fare no better. The authors doubt whether there are practical advantages of the instantaneous methods. In [Armangué *et al.* 02] the method is compared to other linear methods based on the instantaneous epipolar constraint. All methods provide biased estimates due to the linearization.

Discrete motion. In this paragraph the focus is on two-view discrete motion methods. The multiple view methods are the topic of the next paragraph. Two view methods are mostly based on the epipolar geometry. Longuet-Higgins was one of the first who proposed a method for the estimation of the fundamental matrix [LonguetHiggins 81]. His linear *eight-point algorithm* is the basis for many other algorithms.

With the camera calibration matrix $\mathbf{K}$ known the fundamental matrix $\mathbf{F}$ can be upgraded to the essential matrix $\mathbf{E}$ (see sec. 2.3.1). Then, $\mathbf{E}$ can be decomposed to solve for the ego-motion parameters [Hartley & Zisserman 03]. Alternatively, one may estimate $\mathbf{E}$ directly using the five-point algorithm presented in [Nistér 04] or the newer version [Stewénius *et al.* 06] which is numerically more stable. The advantage over the eight-point algorithm is that it works even if the scene is planar which is a *critical surface* for the eight-point algorithm [Maybank 92]. Both algorithms have got their names from the minimal number of points required to solve for the unknowns. They can process more points getting more accurate estimates. However, they do not constitute maximum likelihood estimates due to the minimization of an algebraic error.

The *gold standard* method [Hartley & Zisserman 03] minimizes a geometric error. This requires the estimation of nuisance parameters in form of corrected correspondences $\bar{\mathbf{x}}_\mathbf{l} \leftrightarrow \bar{\mathbf{x}}_\mathbf{c}$, which satisfy the epipolar constraint exactly, i.e. $\bar{\mathbf{x}}_\mathbf{c}^T \mathbf{F} \bar{\mathbf{x}}_\mathbf{l} = 0$. The sum of all squared *reprojection errors*:

$$\sum_{i=1}^{N} d(\mathbf{x}_{\mathbf{l},i}, \bar{\mathbf{x}}_{\mathbf{l},i})^2 + d(\mathbf{x}_{\mathbf{c},i}, \bar{\mathbf{x}}_{\mathbf{c},i})^2 \tag{4.8}$$

is minimized. Each perfect correspondence $\bar{\mathbf{x}}_\mathbf{l} \leftrightarrow \bar{\mathbf{x}}_\mathbf{c}$ has three degrees of freedom, namely the 3D point to which it triangulates. Thus, there are $3N+5$ parameters in total to estimate. The effort pays off as the result is a maximum likelihood estimate, provided that the measured image positions $\mathbf{x}_\mathbf{l}$ and $\mathbf{x}_\mathbf{c}$ are $N(0,\sigma)$ distributed.

Multiple views. Correspondences over more than two frames, if available, impose more constraints on the ego-motion parameters leading to more accurate estimates. Usually, multiple views are considered in *structure from motion* (SfM) methods where the aim is to reconstruct the scene. In this case one is mainly interested in the reconstructed 3D points rather than in the ego-motion. The reconstruction quality benefits if the 3D points are seen from several different points of view. The typical error metric which is minimized is the reprojection error as in the gold standard method but this time applied to F views:

$$\sum_{j=1}^{F}\sum_{i=1}^{N} d(\mathbf{P}_j \bar{\mathbf{x}}_{\mathbf{w},i}\,,\,\mathbf{x}_{i,j})^2 \tag{4.9}$$

The unknowns are the 3D points $\bar{\mathbf{x}}_{\mathbf{w},i}$ and the projection matrices $\mathbf{P}_j = \mathbf{K}\mathbf{R}_j[\mathbf{I}|\mathbf{t}_j]$ (where $\mathbf{P}_1 = \mathbf{K}[\mathbf{I}|\mathbf{0}]$) covering the ego-motion of each frame. There are $3N + 6(F-1) - 1$ parameters in total to estimate. The -1 accounts for the free overall scale factor. It may be fixed by setting the driven distance of the second camera to unity: $||\mathbf{t}_2|| = 1$. Then, the other driven distances $||\mathbf{t}_j||\ j \in [3,F]$ are normal paramters and must be estimated. Hence, we have zero parameters for the first camera, five parameters for the second and six parameters for every additional camera. The algorithm minimizing the cost function 4.9 is known to as *bundle adjustment*. Due to the high number of parameters and the intrinsic non-linearity the algorithm is computationally expensive. Many endeavors have been made to develop efficient implementations.

Bundle adjustment involves the formulation of a large scale, yet sparse, minimization problem. [Engels *et al.* 06] exploits the block diagonal (sparse) form of the Jacobian matrix of the error metric. The Jacobian matrix is used within the Levenberg-Marquardt (LM) minimization. On a 3.4GHz Xeon processor one iteration of LM requires just 0.5ms ($F = 7, N = 260$).

Lourakis and Argyros [Lourakis & Argyros 05] compare the Levenberg-Marquardt minimization to Powell's dog leg minimization. The main advantage of the latter one is that it requires less computations of the Gauss-Newton update: $\delta = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{g}$ where $\mathbf{J}$ is the Jacobian and $\mathbf{g}$ the gradient of the error metric. The solution of this linear equation system is costly if it is high dimensional, which is the case for the bundle-adjustment problem. The dog leg algorithm is 2.0 to 7.0 times faster than LM depending on the number of parameters.

The mobile robotics community is faced with a problem related to SfM: One of the tasks of a mobile robot is to localize itself within its working environment. This requires the knowledge about the environment (scene structure or landmarks[2]). However, the environment might be unknown to the robot. Another task of the robot is to explore the environment while moving trough it. Once, the environment is learned the robot should keep it in mind, so that the robot can immediately re-localize itself after it has been kidnapped or switched off and on. Typically, the working environment is too large to have it entirely in the robots's field of view. Hence, it must store everything it has seen so far in a map and it must be able to associate the data in the map with the data currently present in the field of view. The map building and localization are continuous processes. In contrast to SfM the images cannot be processed as a whole. They must

[2]Landmarks are distinctive objects in the physical world, for example corners of buildings or traffic signs.

be processed recursively. Algorithms solving these tasks are referred to as *online simultaneous localization and mapping* (online SLAM) [Thrun *et al.* 05].

SLAM is a probabilistic framework abstracting from specific sensor technologies. The online version of SLAM estimates the a posteriori probability of the current pose $\mathbf{x}_t = (x, y, z, \alpha, \psi, \varphi)^T$, relative to a fixed coordinate frame, along with the map m, where m is either a list of landmarks (*feature-based*), or a discretized grid of the 3D world (*location-based*). The a posteriori probability is a function of all sensor measurements $\mathbf{z}_{1:t}$ and all controls $\mathbf{u}_{1:t}$. The subscript $1{:}t$ denotes the complete history from time instant 1 up to t. In summary, the task is to estimate:

$$p(\mathbf{x}_t, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{4.10}$$

Where is the ego-motion hidden in this probability? The most likely current pose is the one which maximizes the a posteriori probability:

$$\hat{\mathbf{x}}_t = \arg\max_{\mathbf{x}_t, m} p(\mathbf{x}_t, m \,|\, \mathbf{z}_{1:t}, \mathbf{u}_{1:t}) \tag{4.11}$$

The very recent pose $\hat{\mathbf{x}}_{t-1}$ is computed in the same way. The ego-motion directly follows from both poses. The map m comprising estimated 3D points is a nuisance parameter. Although a lot of algorithms estimating p have been proposed - examples are Extended Kalman filter SLAM and FastSLAM 2.0 (see [Thrun *et al.* 05] for details) - SLAM is still a highly active field of research, as the recent conferences on robotics (`www.iros2006.org`, `www.icra07.org`) indicate.

When the SLAM problem is tackled utilizing a camera, the devised algorithms are referred to as *visual SLAM*. One representative is [Silveira *et al.* 07]. This direct method assumes the imaged scene to be locally planar. The structure parameters (normal vector plus distance of the plane) of each image patch (feature) are estimated along with the ego-motion parameters utilizing the SSD error metric. Additonally, affine illumination changes are modelled for each patch. The estimates are fed into a Kalman filter fulfilling the needs of online SLAM.

While [Silveira *et al.* 07] only compares the image patches (more precisely their grey value structures) over the last three frames, [Molton *et al.* 04] compares the image patch in the first frame to that in the current frame exploiting larger driven distances. An Extended Kalman filter predicts the appearance of an image patch in the current frame based on the observations in the past. The template patch (= patch in the first frame) is pre-warped according to that prediction. It is then matched with the actual observed patch in the current frame. This approach provides more accurate results, especially reducing the drift of the patch's position.

4.3 Motivation

In the last section we have seen that a lot of ego-motion estimation methods exist in the literature. The question is: which method is the most suitable for our needs? What are our needs?

- The camera displacement between consecutive frames may be large, due to a high speed of the ego-vehicle and / or a low frame rate. This causes optical flow vectors of a high magnitude. Hence the instantaneous motion field does not apply.

- The algorithm used for the computation of the optical flow (section 3.2.1) does not track features over time, so multiple view methods are ruled out.
- The algorithm must run in real-time. Methods estimating nuisance parameters are therefore problematic.

As a consequence of these needs we concentrate on two-view discrete motion methods which forbear from the estimation of nuisance parameters. Looking at the literature we find out that the methods based on the epipolar geometry meet our needs.

4.4 Parameterization

Due to the fact that only the translational direction is determinable, a representation of the translation in polar coordinates makes sense. Figure 4.2 illustrates the camera coordinate frame and the Euclidean transformation between two views. We assign specific names to the entities

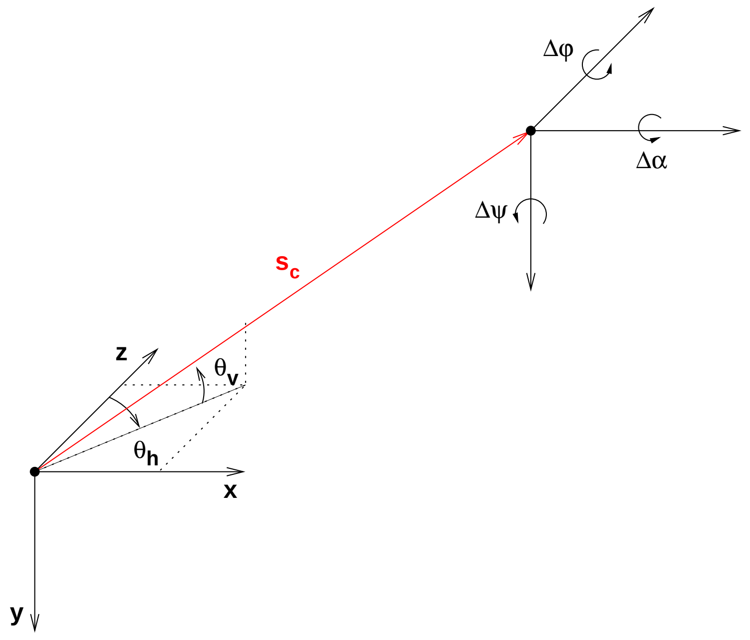


Figure 4.2: Euclidean transformation of the camera (ego-motion) between two frames.

involved:

- rotation about the x-axis: pitch rate[3] $\Delta\alpha$
- rotation about the y-axis: yaw rate $\Delta\psi$

[3]Although the term *rate* is commonly used to express a temporal derivative, we use it here to express differences from frame to frame.

- rotation about the z-axis: roll rate $\Delta\varphi$
- horizontal translational direction θ_h
- vertical translational direction θ_v
- driven distance s_c

The translations along the coordinate axes (t_x, t_y, t_z) are related to the polar coordinate representation as follows:

$$\begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \mathbf{R}(-\theta_v, -\theta_h, 0) \begin{pmatrix} 0 \\ 0 \\ s_c \end{pmatrix} \tag{4.12}$$

with $\mathbf{R}(\alpha_x, \alpha_y, \alpha_z)$ being a rotation matrix with Euler angles in the order z, y, x, see also appendix A. The five ego-motion parameters we can estimate are summarized in the parameter vector $\mathbf{p_e} = (\Delta\alpha, \Delta\psi, \Delta\varphi, \theta_h, \theta_v)$.

4.5 Error Metric

The ego-motion parameters are estimated utilizing the epipolar geometry. In section 2.3.1 we have seen that a correspondence $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$ satisfies the epipolar constraint: $\mathbf{x_c}^T \mathbf{F} \mathbf{x_l} = 0$. The fundamental matrix $\mathbf{F}$ depends on the ego-motion parameters:

$$\mathbf{F}(\mathbf{p_e}) = \mathbf{K}^{-T} [-\mathbf{R}\mathbf{t}]_\times \mathbf{R} \mathbf{K}^{-1} \tag{4.13}$$

with $\mathbf{R} = \mathbf{R}(\Delta\alpha, \Delta\psi, \Delta\varphi)$ and $\mathbf{t} = \mathbf{R}(-\theta_v, -\theta_h, 0)(0,0,1)^T$. Note, that in $\mathbf{t}$ the driven distance s_c is implicitly set to one. The incorporation of s_c would not influence $\mathbf{F}$, since $\mathbf{F}$ is a homogeneous entity. Given at least five correspondences one may minimize the deviations from the epipolar constraint to find an estimate $\hat{\mathbf{p}}_\mathbf{e}$:

$$\hat{\mathbf{p}}_\mathbf{e} = arg \min_{\mathbf{p_e}} \sum_{i=1}^{N} J(\mathbf{F}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}) \tag{4.14}$$

with $J(\mathbf{F}, \mathbf{x_l}, \mathbf{x_c}) = \mathbf{x_c}^T \mathbf{F} \mathbf{x_l}$. However, J is an algebraic error providing biased estimates. An error metric representing a geometric error is the *symmetric epipolar distance* (SED):

$$J_{SED} = \frac{\left(\mathbf{x_c}^T \mathbf{F} \mathbf{x_l}\right)^2}{(\mathbf{F}\mathbf{x_l})_1^2 + (\mathbf{F}\mathbf{x_l})_2^2} + \frac{\left(\mathbf{x_c}^T \mathbf{F} \mathbf{x_l}\right)^2}{(\mathbf{F}^T\mathbf{x_c})_1^2 + (\mathbf{F}^T\mathbf{x_c})_2^2} \tag{4.15}$$

J_{SED} measures the squared distances of the image points to their corresponding epipolar lines: $d^2(\mathbf{x_c}, \mathbf{F}\mathbf{x_l}) + d^2(\mathbf{x_l}, \mathbf{F}^T\mathbf{x_c})$. It provides estimates close to the optimal gold standard method [Faugeras & Luong 01, Hartley & Zisserman 03].

We use this error metric for the estimation but it is not robust yet. In case of ego-motion estimation we are faced with two types of outliers. First, mismatched correspondences and second, correspondences on independently moving objects (IMO). In section 2.4 we have dealt with robust estimation. When a robust estimation method is to be selected the expected amount of outliers have to be guessed.

The fraction of mismatched correspondences have been investigated in section 3.2.1. It was about 6%. The outlier fraction due to an IMO depend mainly on its size in the image, which is large when the IMO is close to the camera. There is one fact which helps reducing this fraction: In traffic scenes IMO's do not suddenly appear direct in front of the ego-vehicle. Instead they are only seen partially when they enter the field of view. Or they start small and get larger when the ego-vehicle is approaching them. Once the IMO's are detected they should be tracked. This allows the exclusion of these image regions from the ego-motion estimation. Thus, IMO's are only outliers as long as they are not detected.

The expected amount of outliers is minor, so the M-estimation is the method of choice. We employ the Huber cost function (equation 2.41) in its "rooted" form since J_{SED} is already squared:

$$C(r) = \begin{cases} r & , |r| < T^2 \\ \sqrt{2T|r| - T^2} & , |r| \geq T^2 \end{cases} \tag{4.16}$$

Finally, the robust estimate is given by:

$$\hat{\mathbf{p}}_{\mathbf{e}} = arg \min_{\mathbf{p_e}} \sum_{i=1}^{N} C(J_{SED}) \tag{4.17}$$

The efficient iterative minimization of this error metric is addressed in section 4.7. The estimated parameters are used as an initial guess for the next frame.

4.6 Motion Model of the Camera

Since we know that the camera is mounted in a vehicle the camera undergoes a restricted motion. We model this motion which reduces the degrees of freedom and makes the estimated ego-motion more stable and accurate. The motion model has also been published in [Klappstein *et al.* 06a].

4.6.1 Horizontal Translational Direction and the Circular Motion Constraint

In this section we consider the steering of the ego-vehicle. Within a small time period (the time between two frames, typically 40 ms) the yaw angle can be approximated as constant. During that time the ego-vehicle on the road drives along a circular arc, i.e. it fulfills a planar circular motion. If the camera is mounted at the rear axle of the ego-vehicle the horizontal translational direction is independent of the driven distance (see figure 4.3). However, it is influenced by the driven angle (yaw rate). Figure 4.4 illustrates this. The yaw rate rotates the horizontal translational direction away from the z-axis. It rotates by an angle which is half the yaw rate angle. This is computed via the isosceles triangle $\mathbf{c_1}\ \mathbf{c_2}\ \mathbf{m}$ and the sum of the inner angles of a triangle:

$$\theta'_h = 90^\circ - \theta_h \tag{4.18}$$

$$\theta'_h + \theta'_h + \Delta\psi = 180^\circ \tag{4.19}$$

$$\theta_h = \frac{1}{2}\Delta\psi \tag{4.20}$$

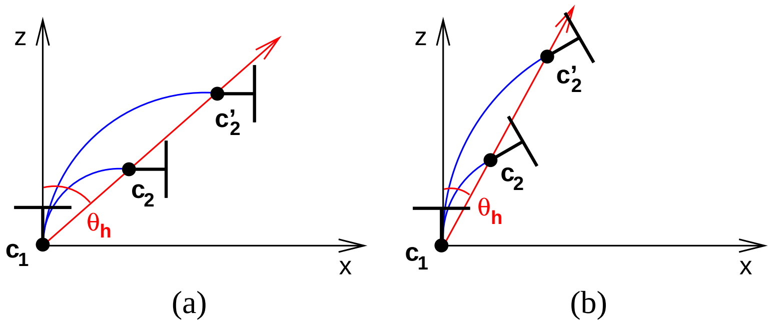


Figure 4.3: Planar circular movement. (a) The camera moves once from c_1 to c_2 and once to c'_2 while rotating around 90° each time. The horizontal translational direction (red line) parametrized with the angle θ_h is identical in both cases. (b) The same consideration with a rotation angle of 60°.

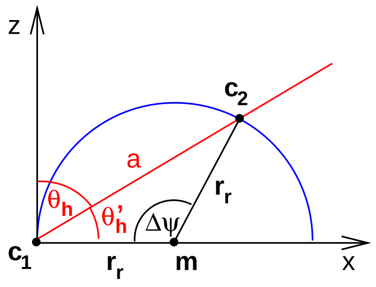


Figure 4.4: Geometric relations of the horizontal translational direction (red line) and the yaw rate $\Delta\psi$ of the camera. $\theta_h = \frac{\Delta\psi}{2}$

Now we consider the more common case where the camera is mounted somewhere in front of the car. Here $\theta_h = \frac{1}{2}\Delta\psi + \beta$ where β is the *kinematic side slip angle*[4]. It depends on the distance of the camera to the rear axle and on the radius of the curvature. For simplicity the Ackermann model [Zomotor 91] is used here (center of gravity lies on the road, no longitudinal forces). The model allows to combine the two wheels into one wheel in the middle of the axle. Further we consider the stationary steering with no side slip. The latter one only holds for small lateral accelerations. Under all these conditions the vehicle dynamic is modelled as illustrated in figure 4.5a.

[4]Following the definition of the term "side slip angle", it applies only at the center of gravity of the vehicle. Here it is used also at the location of the camera.

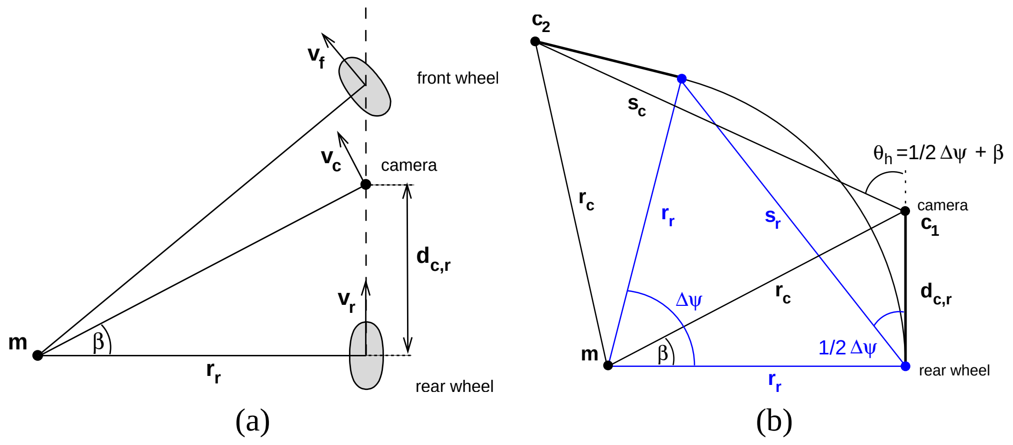


Figure 4.5: Ackermann model and the driven distance of the camera. (a) The camera's velocity v_c and the velocity of the rear axle v_r are different. The angle between these two is β. (b) The driven distance of the camera s_c can be computed if the yaw rate $\Delta\psi$ and the angle β were measured (see text for details).

The velocity of the rear axle v_r is parallel to the vehicle's longitudinal axis (dashed line) and the velocity of the front axle v_f shows to the same direction as the front wheel. The intersection point of the lines orthogonal to v_r and v_f forms the center m of the circle. The radius of curvature at the rear axle is r_r. The camera is mounted at the distance $d_{c,r}$ w.r.t. the rear axle. This arrangement lets the camera's velocity v_c rotate by the angle β:

$$\beta = \arctan \frac{d_{c,r}}{r_r} \tag{4.21}$$

4.6.2 Determining the Scale Factor

In this section an interesting idea is presented how to determine the driven distance (scale factor) utilizing the knowledge about the distance $d_{c,r}$ of the camera to the rear axle.

The fact that β depends on s_c can be exploited to determine s_c, meaning to resolve the scale ambiguity. The relevant geometry is depicted in figure 4.5b. While the rear axle's circular motion has a radius of r_r the camera's radius r_c is slightly larger:

$$r_c = \sqrt{d_{c,r}^2 + r_r^2} \tag{4.22}$$

The driven distance of the camera is given by:

$$s_c = 2\, r_c \sin \frac{\Delta\psi}{2} \tag{4.23}$$

Substituting r_r in equation 4.22 with the equation 4.21 and putting r_c into equation 4.23 results in:

$$s_c = 2\, d_{c,r} \sqrt{1 + \frac{1}{\tan^2 \beta}} \sin \frac{\Delta\psi}{2} \tag{4.24}$$

In order to get an imagination of the required accuracy of β the driven distance is plotted against β as shown in figure 4.6. Thereby, realistic values for the camera displacement to the rear axle and for the yaw angle between two consecutive frames are chosen. For smaller β's (larger radii of curvature) the gradient becomes larger, which means that small errors in β have a more and more severe influence on s_c.

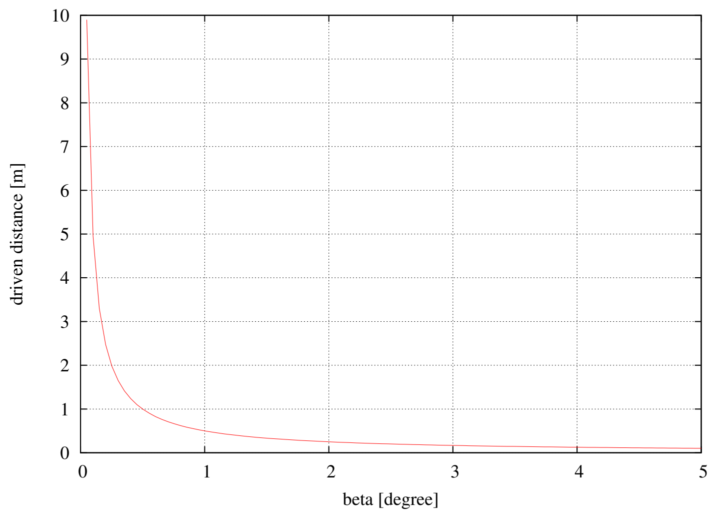


Figure 4.6: The driven distance of the camera s_c against β. The distance was set to $d_{c,r} = 2m$ and the yaw rate to $\Delta\psi = 0.5°$.

Next the relative error of β in dependence on the relative error of s_c is computed in order to derive the required relative accuracy of β. A measurement error $\Delta\beta$ at a specific true value $\bar{\beta}$ causes an error in s_c: $\Delta s_c = s_c(\bar{\beta}+\Delta\beta) - s_c(\bar{\beta})$. The relative error of s_c is:

$$e_s = \frac{\Delta s_c}{s_c(\bar{\beta})} = \frac{s_c(\bar{\beta}+\Delta\beta) - s_c(\bar{\beta})}{s_c(\bar{\beta})} \tag{4.25}$$

Applying the Taylor series expansion up to first order: $s_c(\bar{\beta}+\Delta\beta) \approx s_c(\bar{\beta}) + s_c'(\bar{\beta}) \cdot \Delta\beta$ along with the relative error $e_\beta = \Delta\beta/\bar{\beta}$ yields:

$$e_s = \frac{s_c'(\bar{\beta}) \cdot \bar{\beta}}{s_c(\bar{\beta})} \cdot e_\beta \tag{4.26}$$

Substituting equation 4.24 into 4.26 we get:

$$e_s = -\frac{\bar{\beta}}{\tan\bar{\beta}} \cdot e_\beta \tag{4.27}$$

From $\tan\beta \approx \beta$ for $\beta \ll 1$ it follows: $e_\beta = -e_s$.

We now give a little example to point out a realistic required accuracy: Let's assume that the desired relative accuracy of the driven distance is 5 % $(e_s = 0.05)$. Then, the required relative accuracy of β is about 5 % too. Let's say the largest radius of curvature is $r_r = 230m$. This corresponds to $\beta = 0.5°$ when $d_{c,r} = 2m$. Then the required accuracy of β is $0.025°$! Note, that we cannot estimate β directly, but the horizontal translational direction θ_h. In section 4.8 we will find out that the accuracy of θ_h is much less than the required one.

It is a rather freaky idea to determine the driven distance this way. It works only in curves and the required accuracies are very high. We will strike another more promising path: the estimation of the road homography.

4.6.3 Vertical Translational Direction

The horizontal translational direction is linked to the yaw rate as seen in the last section. Can we also find a link between the vertical translational angle and the pitch rate?

Strictly speaking, no! The reason is that, in contrary to the horizontal direction, the pole of rotation is not constant. This is illustrated in figure 4.7. Two examples of pitch motion are

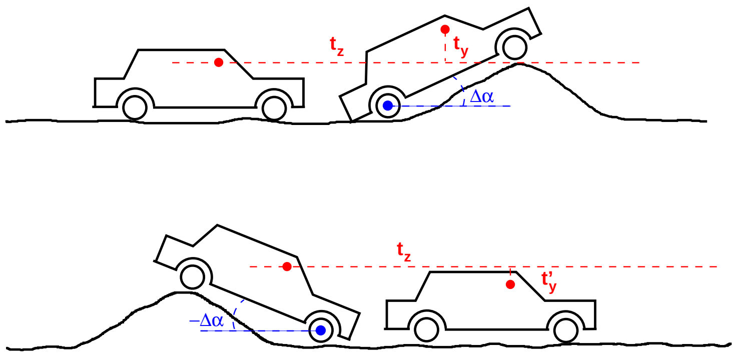


Figure 4.7: Geometric relations of the vertical translation t_y and the pitch rate $\Delta\alpha$ of the camera. Two motion examples are shown demonstrating that t_y is not directly linked to $\Delta\alpha$. Although t_z and $\Delta\alpha$ are constant $t_y \neq t'_y$. Note, that the situation is exaggerated for better visualization.

shown. In the first one the ego-vehicle drives up a hill (bumpy road). It pitches about the angle $\Delta\alpha$ whereas the pitching pole is the rear axle. This causes a vertical translation t_y of the camera. In the second example the ego-vehicle drives down a hill. The pitch angle is the same but the pitching pole is now the front axle. This causes a vertical translation t'_y which is different from t_y. Consequently, there is no direct link between the vertical translational direction θ_v and the pitch rate $\Delta\alpha$. Without knowing the pitching pole a correct modelling of θ_v is infeasible.

We, nevertheless, model θ_v. We just pretend that the height of the camera above the road does not change, i.e. t_y is clamped to zero and the pitching pole is assumed to coincide with the camera center. Under these assumptions the vertical translational direction is equal the pitch angle of the road w.r.t. the camera: $\theta_v = \alpha$. Due to pitch motions of the ego-vehicle the pitch angle changes continuously. However, the current pitch angle cannot be estimated from the optical flow alone. Only rotations from frame to frame, i.e. the pitch rate $\Delta\alpha$, can be estimated. If the road has a constant vertical slope as in figure 4.8a the pitch rate is identical to the temporal derivative of the absolute pitch angle α. Consequently, α may be retrieved through integration:

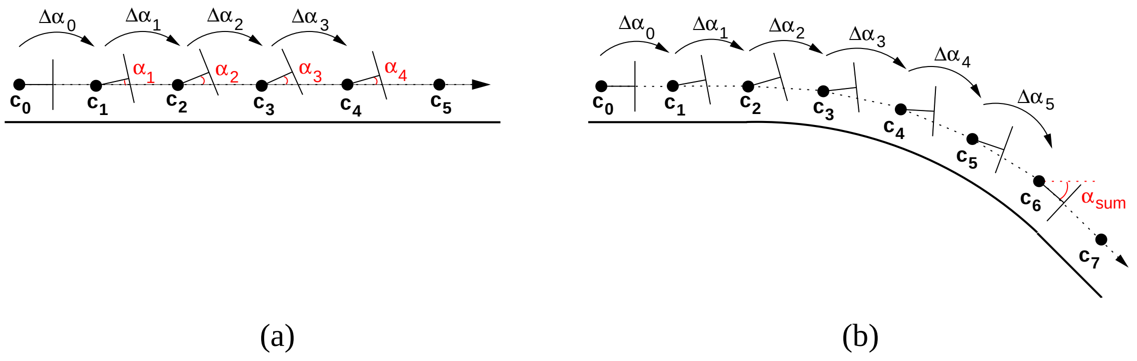


Figure 4.8: Absolute pitch angle in relation to the pitch rates. (a) The road has a constant vertical slope. The pitch angle α is the sum of the pitch rates $\Delta\alpha_i$. (b) The vertical slope of the road is changing. The actual pitch angle is totally different from the summed pitch rates.

$$\alpha_i = \alpha_0 + \sum_{i=0}^{N} \Delta\alpha_i \tag{4.28}$$

where α_0 is the initial pitch angle. It can be determined through offline calibration. Alternatively, one can exploit the fact that the long term average of the pitch rate must be zero, otherwise the ego-vehicle would loop the loop (see section 4.8 accuracy of ego-motion).

One drawback of this pitch integration approach is that it is invalid if the vertical slope of the road changes as illustrated in figure 4.8b. The integrated pitch rates do not reflect the absolute pitch angle. Another severe drawback is the feedback introduced by setting $\theta_v = \alpha$, depicted in figure 4.9. The current pitch rate estimation depends on the recent estimated pitch rates. If the

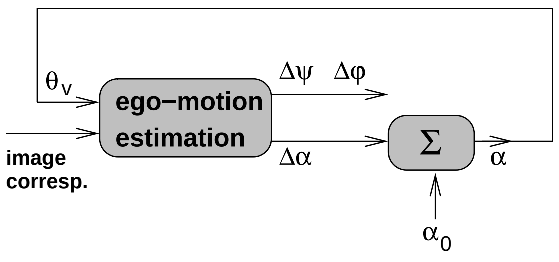


Figure 4.9: Feedback within in the motion model based ego-motion estimation.

initial angle α_0 is set too high (or too low) the pitch rate estimates are too high (or too low) as well. This is an amplifying effect. The estimation error increases (or decreases) in each frame. Thus, the entire estimation process is a labile equilibrium.

To confirm this experimentally we estimate the ego-motion using a highway sequence shown in figure 4.10. Figure 4.11a shows the results of the pitch angle for the highway sequence where α_0 was slightly too low. In figure 4.11b α_0 was too high. Note, that the difference for the α_0's in both figures is just 0.01°. The instability becomes apparent. The feedback in this way does not work. In chapter 5 we will estimate the road homography which will give us an estimate of the absolute pitch angle. We will use it to refine the feedback.

Figure 4.10: The sequence used to investigate the estimation of the pitch angle. A highway was chosen which has a constant vertical slope. The installation pitch angle was obtained by calibration which was $\alpha_0 = -4.66°$. (a) First frame of the sequence. (b) Frame 100 of the sequence.

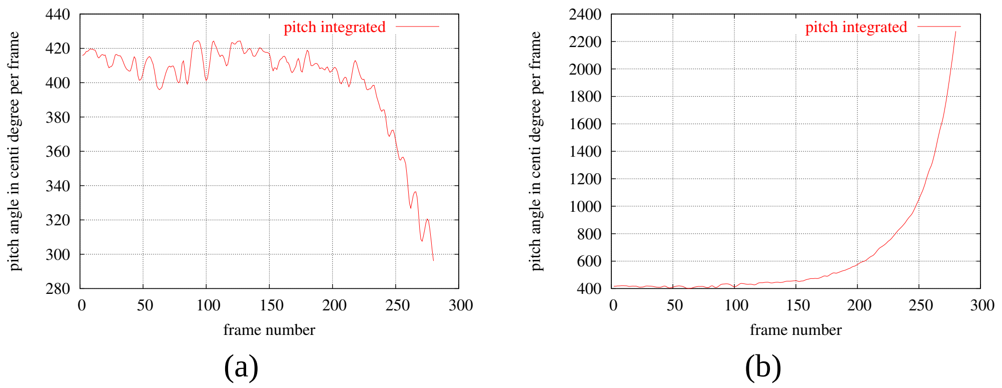


Figure 4.11: Integrated pitch angle α of the highway sequence. (a) The initial angle α_0 was set to $-5.15°$ which was too low. (b) α_0 was set to $-5.14°$ which was too high.

4.6.4 Rolling

One might think that the rolling of the vehicle is negligible. But experiments have shown that a considerable roll rate exists in the real world. Notice that the ego-motion parameters describe the motion of the ego-vehicle with respect to the world, not to the road. So, the roll rate will be different from zero, if the road itself rolls. An example is shown in figure 4.12.

If the roll rate is not estimated (set to zero) the other rotational parameters are influenced by the effects of the roll rate which leads to wrong estimates. Therefore the rolling must be included in the estimation.

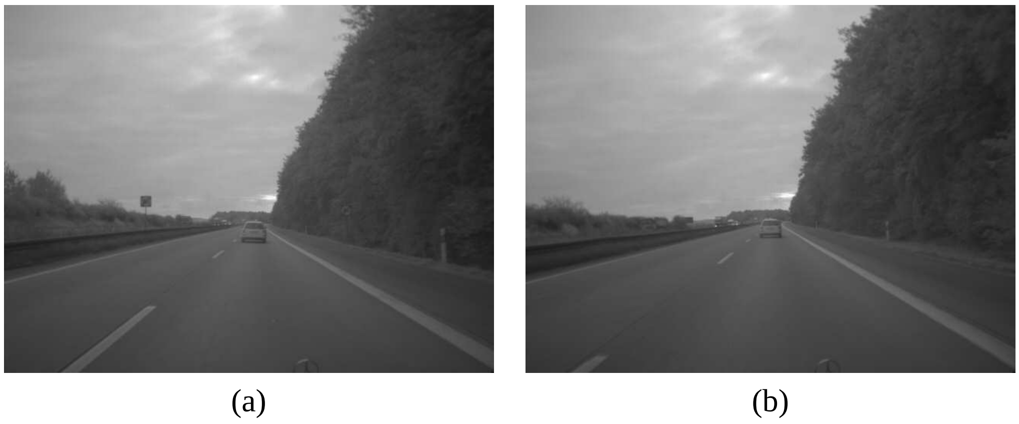


Figure 4.12: Highway sequence with present roll rate. (a) Frame 240 (b) Frame 270. The vehicle has driven approximately 35 meters and it rolled around 3°.

4.6.5 Summary

The motion of the camera is modelled as planar and circular. The horizontal translational direction θ_h is a function of the pitch rate $\Delta\alpha$ and the driven distance s_c. The latter one is either retrieved by odometry or by the estimated road homography. The vertical translational direction θ_v is clamped to the pitch installation angle α_0 obtained by calibration. The motion model reduces the ego-motion parameters to the rotational ones: $\mathbf{p_e} = (\Delta\alpha, \Delta\psi, \Delta\varphi)$.

4.7 Efficient Minimization

As discussed in section 4.5 the solution of the ego-motion problem is given by:

$$\hat{\mathbf{p}}_\mathbf{e} = arg \min_{\mathbf{p_e}} \sum_{i=1}^{N} C(J_{SED}) \tag{4.29}$$

Due to the non-linearity of $C(J_{SED})$ the solution $\hat{\mathbf{p}}_\mathbf{e}$ must be found by an iterative minimization. The time spent for this minimization is the crucial point for the real-time capability. In this section we study three minimization schemes of different types. The first uses only the function itself. The second takes advantage of the gradient, and the last uses the Hessian matrix in addition.

The Powell algorithm [Press *et al.* 02], a gradient free descent approach, efficiently minimizes quadratic functions. Note that near a minimum any function is approximately quadratic (Taylor series expansion up to second order).

$C(J_{SED})$ is quadratic in a relative large area around the minimum. Figure 4.13 shows the graph of $C(J_{SED})$ when varying the yaw rate. The other two parameters (pitch and roll rate) are kept constant in the minimum. The cuts of $C(J_{SED})$ through the other parameters are similar. They are not shown here.

Given an N-dimensional quadratic function Powell needs at least $N \cdot (N+1) \cdot 3$ function calls to find the minimum. N iterations are required to establish the optimal (conjugate) search

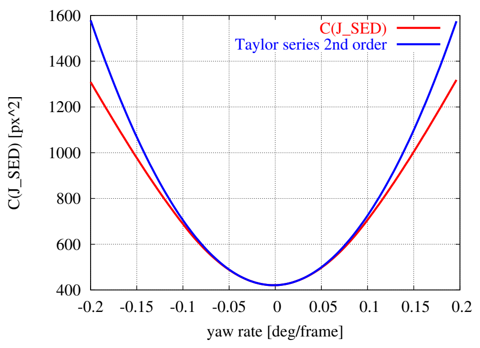


Figure 4.13: Cut of $C(J_{SED})$ through the yaw rate. The Taylor series expansion up to second order of (J_{SED}) (blue line) shows that $C(J_{SED})$ is almost quadratic in a wide range around the minimum.

directions. In each iteration the function is minimized along $N+1$ one-dimensional directions (lines). Every line minimization is performed by a parabolic fit requiring 3 function calls.

When the gradient and the Hessian matrix of the function are available, the minimum is found in a single step, called Newton step. The complexity in doing so is $1+N+N\cdot(N-1)$. This is 1 function call, N calls of the first partial derivatives, and $N\cdot(N-1)$ calls of the second partial derivatives. It is assumed that computing the derivatives is as expensive as computing the function itself.

The error metric $C(J_{SED})$ depends on the rotational parameters, thus we have a three-dimensional minimization task. In such a space Powell needs 36 function calls while the usage of the gradient + Hessian matrix needs only 10 "function calls"[5] to find the minimum of a quadratic function.

Thus computing the first and second derivatives saves a lot of time. By doing this another fact shortens the computation time: Every single function call requires the CPU to load the correspondences into its registers. This is very expensive if the correspondences are not cached. Less function calls reduce the amount of cache misses.

The usage of the gradient and Hessian matrix requires a minimization scheme which handles this information. HUMSL (Hessian provided Unconstrained Minimization SoLver) [Gay 83] is such a scheme. It is available under `www.netlib.org/port`.

The Powell and the HUMSL algorithm are designed to minimize an arbitrary function. When the function is based on least squares, i.e.:

$$\chi(\mathbf{p}) = \sum_i r(\mathbf{p})_i^2 \tag{4.30}$$

with $r(\mathbf{p})_i$ the individual residuals and $\mathbf{p}$ the parameter vector, special minimization schemes can be applied to find the minimum over $\mathbf{p}$. A very famous one is the Levenberg-Marquardt (LM) minimization [Press *et al.* 02]. One key idea of LM is an abbreviation in the computation of the Hessian matrix. To see this we compute the first derivative of 4.30 with respect to the k-th

[5]The term "function call" here subsumes the actual function call and the call of the derivatives.

parameter:

$$\frac{\partial \chi}{\partial (\mathbf{p})_k} = 2 \sum_i r_i \frac{\partial r}{\partial (\mathbf{p})_k} \tag{4.31}$$

The second derivative with respect to the k-th and l-th parameter then reads:

$$\frac{\partial^2 \chi}{\partial (\mathbf{p})_k \partial (\mathbf{p})_l} = 2 \sum_i \frac{\partial r_i}{\partial (\mathbf{p})_l} \frac{\partial r_i}{\partial (\mathbf{p})_k} + r_i \frac{\partial^2 r_i}{\partial (\mathbf{p})_k \partial (\mathbf{p})_l} \tag{4.32}$$

The Levenberg-Marquardt algorithm cancels out the second term $r_i \frac{\partial^2 r_i}{\partial (\mathbf{p})_k \partial (\mathbf{p})_l}$ in 4.32. By doing so one assumes that the residuals r_i are zero-mean and uncorrelated. The zero-mean property can only hold in the minimum of $\chi(\mathbf{p})$. Near the minimum the residuals are approximately zero-mean. Beside the assumption of zero-mean and uncorrelated r_i's it is assumed that they are uncorrelated with their second derivatives. When the second derivatives are zero or nearly zero also the second term cancels out. Iterative minimization schemes using these assumptions are known as Gauss-Newton schemes. The minimization of $\chi(\mathbf{p})$ requires just $1+N$ function calls if the assumptions are true. In our three-dimensional space (three rotational DoF) 4 calls are enough.

In most LM implementations the user is requested to put in the residuals r_i. Is it a good idea to use $C(J_{SED})$ as residual r? No, $C(J_{SED})$ represents the squared symmetric epipolar distance, thus is always positive. Further, the second derivatives are far from zero. In order to apply the LM algorithm we have to modify $C(J_{SED})$ slightly. The "rooted" version of J_{SED}:

$$J_{RSED} = \mathbf{x_c}^T \mathbf{F} \mathbf{x_l} \cdot \sqrt{\frac{1}{(\mathbf{F}\mathbf{x_l})_1^2 + (\mathbf{F}\mathbf{x_l})_2^2} + \frac{1}{(\mathbf{F}^T\mathbf{x_c})_1^2 + (\mathbf{F}^T\mathbf{x_c})_2^2}} \tag{4.33}$$

along with the point-symmetric "rooted" Huber cost function:

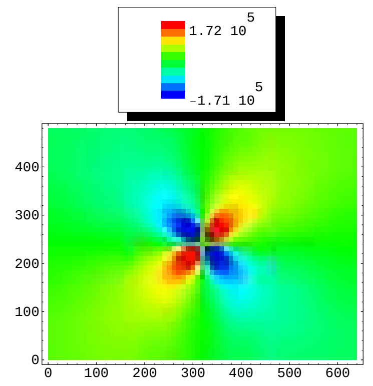
$$C_p(r) = \begin{cases} r & , |r| < T^2 \\ \sqrt{2T|r| - T^2} \cdot \operatorname{sgn} r & , |r| \geq T^2 \end{cases} \tag{4.34}$$

makes the epipolar distance positive or negative depending on which side of the epipolar line the corresponding point lies. The necessary condition for zero-mean residuals is now fulfilled. The consideration of the sufficient condition is postponed until section 5.6.2 where we will prove that under isotropic noise the residuals are actually zero-mean.

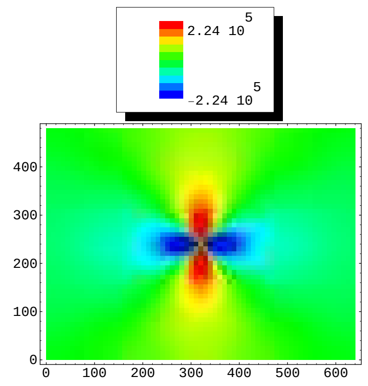
Are the second derivatives of 4.33 zero? No, they depend on the ego-motion parameters and the image position. Figure 4.14 shows the second derivatives in the minimum of J_{RSED} for an ego-motion along the optical axis. Some of the second derivatives are unbounded in the epipole. Image regions near the epipoles $\mathbf{e_l}$ and $\mathbf{e_c}$ are therefore excluded:

$$J_{RSED} = 0 \mid \|\mathbf{x_l} - \mathbf{e_l}\|_\infty \leq 3\text{px} \cup \|\mathbf{x_c} - \mathbf{e_c}\|_\infty \leq 3\text{px} \tag{4.35}$$

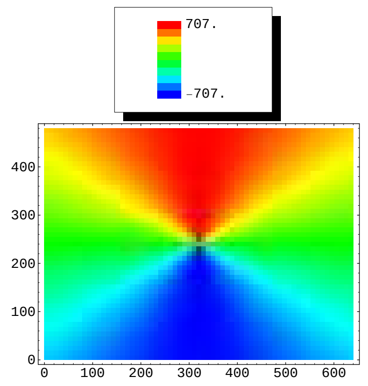
Figure 4.14 shows three out of six second derivatives. The derivatives not shown look similar to the ones shown. Rotating 4.14(a) around 90° yields $\frac{\partial^2 J_{RSED}}{\partial \Delta\psi^2}$ and 4.14(c) rotated around 90° yields $\frac{\partial^2 J_{RSED}}{\partial \Delta\psi \partial \Delta\varphi}$. The second derivative w.r.t. the roll rate $\frac{\partial^2 J_{RSED}}{\partial \Delta\varphi^2}$ is zero in the minimum.



(a)



(b)



(c)

Figure 4.14: Second derivatives of J_{RSED} in dependence of the image position for an ego-motion along the optical axis. (a) $\frac{\partial^2 J_{RSED}}{\partial \Delta\alpha^2}$ (b) $\frac{\partial^2 J_{RSED}}{\partial \Delta\alpha \partial \Delta\psi}$ (c) $\frac{\partial^2 J_{RSED}}{\partial \Delta\alpha \partial \Delta\varphi}$

Near the minimum all second derivatives only change slightly. The figure shows that the second derivatives are not zero in all image regions. However, it points out that the second derivatives are symmetric or point-symmetric relative to the epipole. If the correspondences are uniformly distributed over the image the second derivatives are zero-mean satisfying the LM assumption. Thus, we expect the LM algorithm to be superior over the other two algorithms. In the following this is confirmed experimentally.

Up to now we assumed that computing the first and second derivatives is as expensive as computing the function which makes LM considerably faster (LM: 4 calls, HUMSL: 10, Powell: 36). However, computing the derivatives of $C_p(J_{RSED})$ is cumbersome since the rotation matrix **R** included in the Fundamental matrix **F** comprises products of sine and cosine functions. Luckily for us, the expected rotations are small, thus we can employ the linearized rotation matrix:

$$\mathbf{R}_{\text{lin}} = \mathbf{I} + \left[\begin{pmatrix} \Delta\alpha \\ \Delta\psi \\ \Delta\varphi \end{pmatrix} \right]_{\times} = \begin{bmatrix} 1 & \Delta\varphi & -\Delta\psi \\ -\Delta\varphi & 1 & \Delta\alpha \\ \Delta\psi & -\Delta\alpha & 1 \end{bmatrix} \qquad (4.36)$$

making the derivatives much easier to compute. The mathematical effort for computing $C_p(J_{RSED})$, measured in terms of multiplications, is 22 per correspondence. Each first derivative costs 29 multiplications and each second derivative costs 40 multiplications. There is an additonal cost for the preparation required once per call: composition of the F-matrix and computation of the epipoles: 138 multiplications, and computation of the derivatives of the F-matrix: 301 multiplications.

The average number of function calls required to find the minimum was obtained by experimental tests. They are 102 (Powell), 6.5 (LM[6]), and 5 (HUMSL). The overall mathematical effort is summarized in table 4.1. Compared to Powell, LM is three times faster. But looking at the actual computation times, which were measured on a Pentium IV 2.4 GHz, LM is even four

[6]The implementation is due to Lourakis: `www.ics.forth.gr/˜ lourakis/levmar`

	Powell	LM	HUMSL
usage of	function	function gradient	function gradient Hessian
multipl. per preparation	138	439	439
multipl. per corresp.	22	22 $+3\cdot 29$	22 $+3\cdot 29$ $+6\cdot 40$
avg. number of calls	102	6.5	5
overall effort for 300 corresp.	687276	215404	525695
relative to Powell	100%	31%	76%
computation time	4.5ms	1.1ms	2.1ms
relative to Powell	100%	24%	47%

Table 4.1: Comparison of different minimization schemes.

times faster. This is due to less cache misses. By the way, computing the derivatives numerically using forward differences is not faster than the analytical derivatives.

4.8 Accuracy of the Ego-Motion Estimation

In this section we address the following questions:

- How many correspondences are required to get a good estimate?
- To which extent does the motion model improve the estimate?
- Does a wrong camera installation angle spoil the estimation when using the motion model?
- Does the noise in the optical flow have a high influence on the accuracy?

To answer these questions we carry out the following simulation: A certain number of world points is generated and imaged onto the last and current frame according to some random but known ego-motion. The image points in the last frame are uniformly distributed over the image. The depth z of the world points is distributed as $z \sim \frac{z_{\min}}{G(0,1)}$ where $z_{\min}$ is the minimal depth and $G(0,1)$ is the uniform distribution. This allows world points to lie very far away. The image points in the current frame get an additive noise according to $N(0,\sigma)$. A certain fraction of the points become outliers. To this end the image point in the current frame is set randomly around the image point in the last frame, whereas the distance between the two image points is normal distributed with $N(\mu = 0\text{px}, \sigma = 30\text{px})$.

The ego-motion is estimated 250 times while varying the world points and the ground truth ego-motion each time. The estimated parameters are then compared to the ground-truth parameters. The ranges of the ego-motion parameters in traffic scenes are typically -0.5 .. $0.5\frac{\text{deg}}{\text{frame}}$ for

parameter	sign	value
number of correspondences		100
focal length	$f_x = f_y$	1000px
minimal depth	$z_{\min}$	10m
distance between last and current frame	s_c	1m
noise in the correspondences	σ	0.55px
outlier fraction		0%

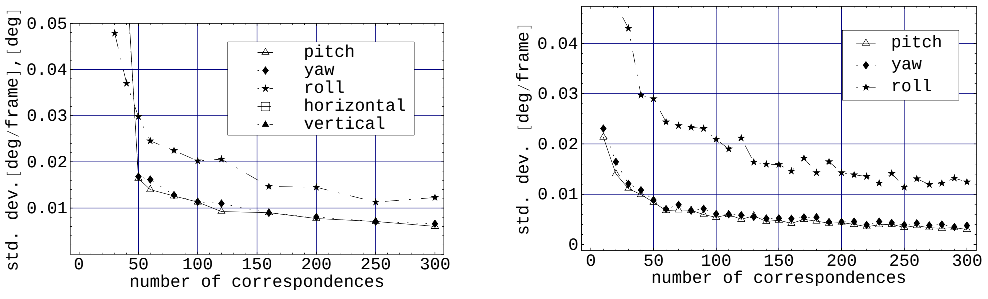
Table 4.2: Parameter values used in the simulation.

the rotational and -20 .. 20 deg for the translational parameters. The ground truth ego-motion is uniformly distributed within these ranges.

The upcoming figures depict the accuracy of the ego-motion estimation in dependence on single parameters. All other parameters are fixed to values shown in table 4.2.

4.8.1 Number of Correspondences

Figure 4.15 shows the standard deviations of the rotational parameters against the number of correspondences. In figure 4.15a all five parameters are estimated. In figure 4.15b the translational



(a) (b)

Figure 4.15: Accuracy of ego-motion estimation depending on the number of correspondences. (a) All five parameters (rotation and translation) are estimated. (b) Only rotations are estimated. The translations are fixed at their actual value.

parameters are known (set to the actual values) and only the rotational parameters are estimated. This is equal to applying the motion model. The usage of the motion model almost doubles the precision of the pitch and the yaw rate. The precision of the roll rate, however, does not benefit from the motion model.

The standard deviation of the estimated parameters increases heavily when less than 50 correspondences are supplied[7]. This is mainly the case if all five parameters are estimated. Figure 4.15a is zoomed out in figure 4.16.

[7]We observe this empirically but we cannot explain the “50”.

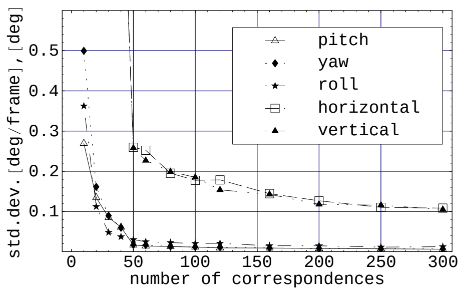


Figure 4.16: Accuracy of ego-motion estimation depending on the number of correspondences. All five parameters are estimated. Using less than 50 correspondences provides poor results.

4.8.2 Minimal Depth

The estimation of the translational parameters is not only more inaccurate than the one of the rotational parameters but also the estimation requires 3D points which are close to the camera. What happens if close 3D points are missing is illustrated in figure 4.17. In traffic scenes it can not be guaranteed that close 3D points are present. Thus, the translational parameters can not be estimated reliably. And when some parameters cannot be estimated reliably why should they be estimated at all? This is another reason to apply the motion model.

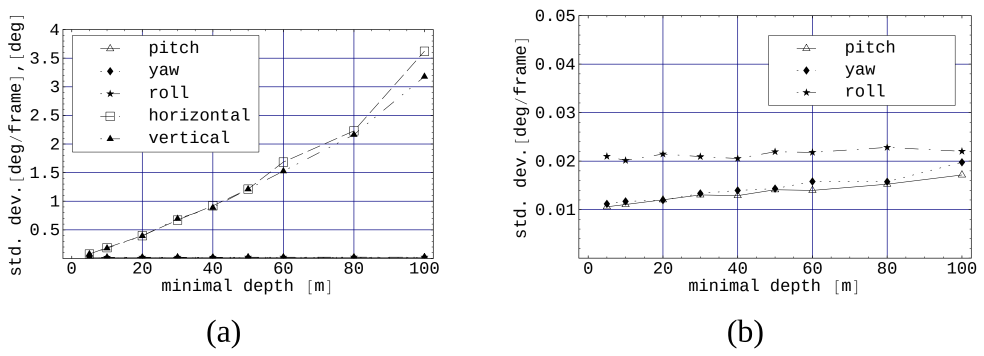


Figure 4.17: Accuracy of ego-motion estimation in dependence on the minimal depth of the 3D-scene. The standard deviation of the translational parameters increases when near 3D-points are missing (a), whereas the rotational parameters are hardly influenced (b).

4.8.3 Deviation from the Motion Model

When the motion model is used, the knowledge about the camera installation angles is required. Commonly this knowledge is obtained by calibration. If the actual angles deviate from the calibration, for example due to a lack of long term stability or pitch motions, the estimated rotational parameters are biased. In figure 4.18 the bias in the pitch rate is shown when the vertical translational direction deviates from the actual one. This bias is equal to the average of the pitch rates, since pitch rotations are zero mean. Otherwise we would loop the loop.

The pitch installation angle α_0 may be calibrated online by observing the average of the pitch rates. However, the bias depends on the scene structure, but the average scene structure is not known. One cannot deduce directly the error of the installation angle from the bias. The true installation angle must be found iteratively. The yaw installation angle is found in the same way as the pitch installation angle, because the yaw motion averaged over a long time period is zero, too. It means in average the ego-vehicle drives straight ahead. The roll installation angle cannot be determined this way.

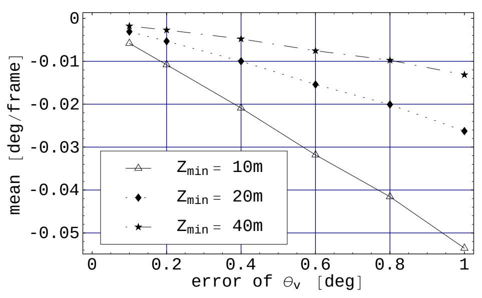


Figure 4.18: Bias in the pitch rate estimation when the assumed vertical translation θ_v is wrong.

4.8.4 Outliers

In the simulations discussed so far the correspondences were free from outliers. In real life outliers occur when the optical flow algorithm produces mismatched correspondences or when independently moving objects (IMO's) are present. Mismatched correspondences are uncorrelated whereas correspondences on IMO's are highly correlated. Here we care about uncorrelated outliers. The generation of outliers is explained at the beginning of this section.

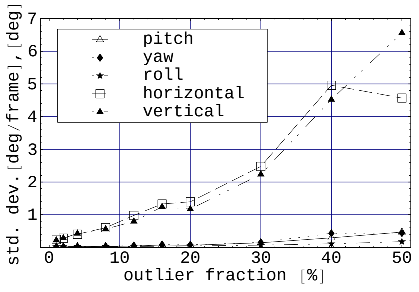
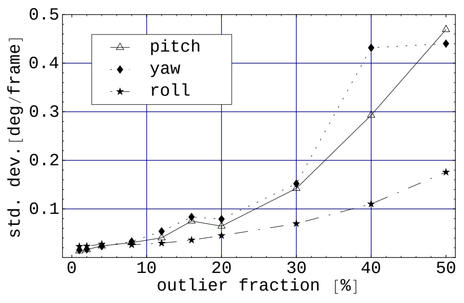
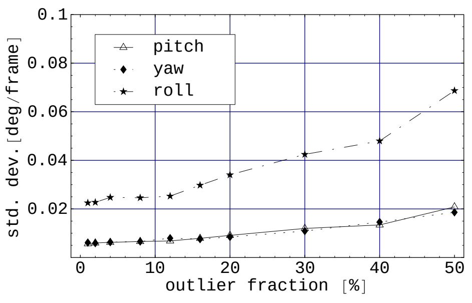


Figure 4.19: Accuracy of ego-motion estimation depending on the outlier fraction. All five parameters are estimated (rotation and translation).

Figure 4.19 shows the accuracy of the ego-motion estimation while the fraction of outliers is varied. As expected, the accuracy becomes worse for higher outlier fractions, especially the translational parameters are influenced heavily by outliers. The application of the motion model



(a)



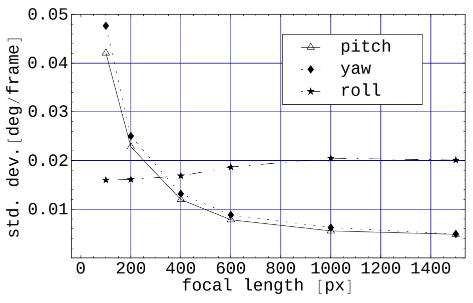
(b)

Figure 4.20: Accuracy of ego-motion estimation depending on the outlier fraction. (a) All five parameters are estimated (rotation and translation). The figure is a close-up of figure 4.19 to point out the rotational parameters. (b) Only rotations are estimated. The translations are fixed to their actual value.

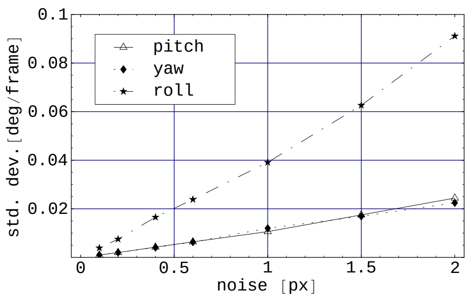
stabilizes the ego-motion estimation to a surprising extent (figure 4.20): The accuracy of the roll rate doubles when the outlier fraction is high ($\geq$30%). The pitch and the yaw rate are already more accurate (factor of 2) when the motion model is applied. If outliers are present in addition the headstart of the accuracy increases further. For example having an outlier fraction of 30% the pitch and the yaw rate are 10 times (!) more accurate compared to the full five DoF estimation. This observation confirms the high correlation between the translational and the rotational parameters. Trying to estimate the translational parameters will result in an overall poor performance.

Ego-motion estimation methods that split the translational parameters from the rotational ones, for instance the series of linear subspace methods originally introduced by [Jepson & Heeger 90], may provide better results than our minimization of the symmetric epipolar distance (SED). However, in the literature there is no extensive study on the achievable accuracy.

The accuracy of the ego-motion parameters is also influenced by the focal length of the camera and by the noise in the optical flow. The graphs are shown in figure 4.21.



(a)



(b)

Figure 4.21: Accuracy of ego-motion estimation depending on (a) the focal length and (b) the noise level.

4.8.5 Summary

A reasonable estimate involves at least 50 correspondences. The translational parameters are inaccurate, especially when close 3D points are missing or when outliers are present. Due to the correlation between translation and rotation the rotational parameters suffer from the poor accuracy of the translational parameters. The motion model breaks the correlation and improves the accuracy of the rotational parameters considerably.

4.9 Sensitivity of the Ego-Motion Parameters

In the last section, we have seen the different accuracies of the ego-motion parameters. We now ask for the reason of these differences.

In general, parameter estimation means minimization of some cost function. The higher the curvature of the error metric in the minimum, the more accurate the estimation will be. If a slight change in the parameter causes a high change in the error metric, due to a high curvature, we say the parameter is sensitive. To get the sensitivities of the ego-motion parameters we compute the second partial derivatives (curvature) of the SED function in the minimum $\hat{\mathbf{p}}_\mathbf{e}$ obtained by equation 4.17:

$$s_{\Delta\alpha} = \left.\frac{\partial^2 J_{SED}}{\partial\Delta\alpha^2}\right|_{\mathbf{p}=\hat{\mathbf{p}}_\mathbf{e}} \tag{4.37}$$

$$s_{\Delta\psi} = \left.\frac{\partial^2 J_{SED}}{\partial\Delta\psi^2}\right|_{\mathbf{p}=\hat{\mathbf{p}}_\mathbf{e}} \tag{4.38}$$

$$s_{\Delta\varphi} = \left.\frac{\partial^2 J_{SED}}{\partial\Delta\varphi^2}\right|_{\mathbf{p}=\hat{\mathbf{p}}_\mathbf{e}} \tag{4.39}$$

$$s_{\Delta\theta_h} = \left.\frac{\partial^2 J_{SED}}{\partial\Delta\theta_h^2}\right|_{\mathbf{p}=\hat{\mathbf{p}}_\mathbf{e}} \tag{4.40}$$

$$s_{\Delta\theta_v} = \left.\frac{\partial^2 J_{SED}}{\partial\Delta\theta_v^2}\right|_{\mathbf{p}=\hat{\mathbf{p}}_\mathbf{e}} \tag{4.41}$$

The sensitivities depend on the image position $\mathbf{x_c}$ and the camera calibration. Furthermore $s_{\Delta\theta_h}$ and $s_{\Delta\theta_v}$ also depend on the depth of the 3D point. Figure 4.22 shows the sensitivities depending on the image position $\mathbf{x_c}$ for a camera motion along the optical axis, i.e. $\hat{\mathbf{p}}_\mathbf{e} = (0,0,0,0,0)^T$. The driven distance is one meter and the focal length is $f_x = f_y = 1000px$. The principal point is set to the center of the image. The depth of the 3D point is $z = 20m$. One clearly sees that not all image regions provide an equal contribution to the accuracy of the ego-motion parameters.

The reason for this is explained in figure 4.23. It shows what happens if the ego-motion parameters move slightly away from their true values. Let's consider a point correspondence $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$ in the image. A slight change of a rotation parameter (figure 4.23a to 4.23c) shifts the original point $\mathbf{x_c}$ to $\mathbf{x}_\alpha$, $\mathbf{x}_\psi$, and $\mathbf{x}_\varphi$ respectively. The new epipolar line $\mathbf{l_l}$ goes through the shifted point and the epipole $\mathbf{e_l}$. The wrong value of the parameter induces an epipolar distance d of the point $\mathbf{x_l}$ to the epipolar line $\mathbf{l_l}$. In the case of the translational parameters (figure 4.23d and 4.23e) a slight change shifts the epipole to $\mathbf{e}_{\theta_h}$ and $\mathbf{e}_{\theta_v}$ respectively. The new epipolar line goes through the shifted epipole and $\mathbf{x_l}$, since a 3D point at infinity is always imaged to the same location, namely $\mathbf{x_l}$, regardless of the translation of the camera.

All figures, 4.23a to 4.23e, contain two correspondences, $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$ and $\mathbf{x_l'} \leftrightarrow \mathbf{x_c'}$, at different image positions to demonstrate the dependence of the sensitivity (distance) on the image position. The primed position $\mathbf{x_c'}$ has a higher distance d' than the non-primed $\mathbf{x_c}$.

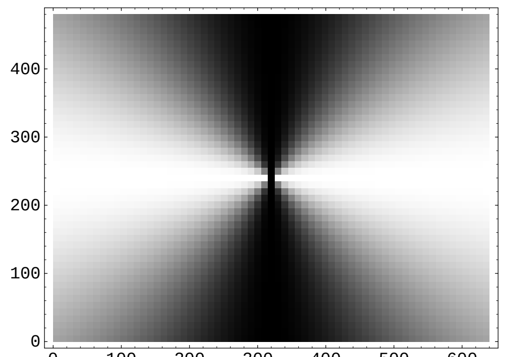
The sensitivity analysis shows that the translational parameters are more insensitive than the rotational ones. The poor results regarding the accuracy we obtained last section (sec. 4.8) are the consequence of this insensitivity.

There is another important point revealed by this analysis: pitch and yaw rotations shift the epipole out of its central position. The sensitive areas (marked white in figure 4.22) go hand in hand with the epipole. Hence, the sensitivity decreases the more the camera pitches or yaws. This is especially the case if the epipole goes outside the image. Strong yaw movements occur at intersections when the car turns into another road. We expect a reduced accuracy in such a case.

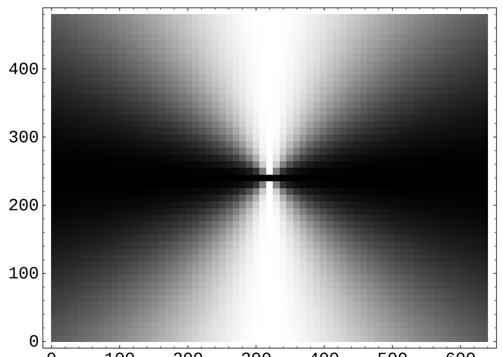
The translational parameters, which are linked to the pitch and to the yaw installation angles, have a quite similar impact on the epipole.

Commonly, the camera looks straight ahead. But when the task is to observe crossing traffic at an intersection the camera must look sidewards. The yaw installation angle then may be 90°. This implies that $\theta_h = 90^\circ$ provided that the yaw rate is zero ($\Delta\psi = 0$). It means the camera moves sidewards. The epipole in that case is at infinity and so are the sensitive areas of the yaw rate. Figure 4.24a shows this. The yaw rate's maximum sensitivity is reduced by a factor of 130 compared to $\theta_h = 0^\circ$. The effect on the accuracy is shown in figure 4.24b.

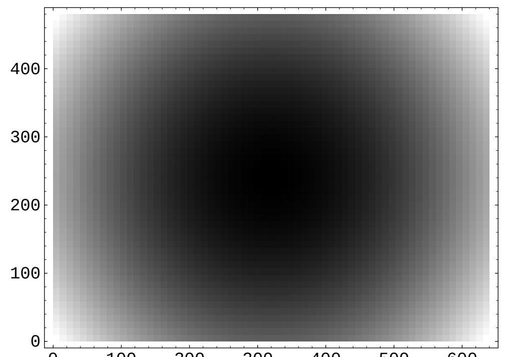
A last point shall be mentioned in this section: we have seen that the sensitivities in some image regions are higher than in others. As a consequence, the accuracy of the estimates is affected by the selection of the correspondences. If correspondences are selected only in low-sensitive regions the estimate will be poor. In order to prevent this, they should be uniformly distributed over the image.



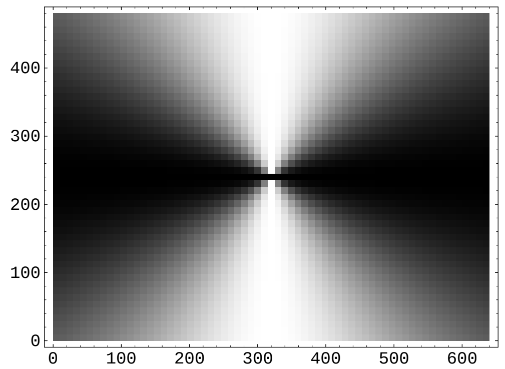
(a)



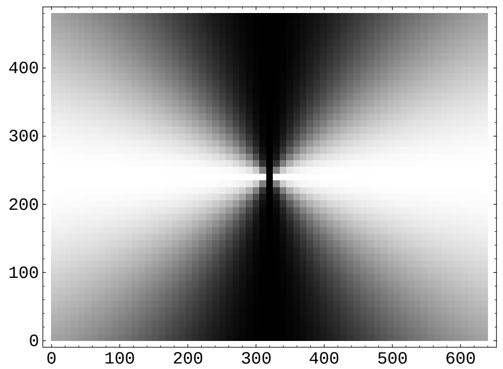
(b)



(c)



(d)



(e)

Figure 4.22: Sensitivities of the ego-motion parameters depending on the image position. Dark regions are regions with low sensitivity. White regions have the maximum sensitivity for that parameter. (a) pitch rate. $s_{\Delta\alpha,max} = 4.43 * 10^6$ (b) yaw rate. $s_{\Delta\psi,max} = 4.43 * 10^6$ (c) roll rate. $s_{\Delta\varphi,max} = 6.4 * 10^5$ (d) horizontal direction. $s_{\Delta\theta_h,max} = 1.1 * 10^4$ (e) vertical direction. $s_{\Delta\theta_v,max} = 1.1 * 10^4$

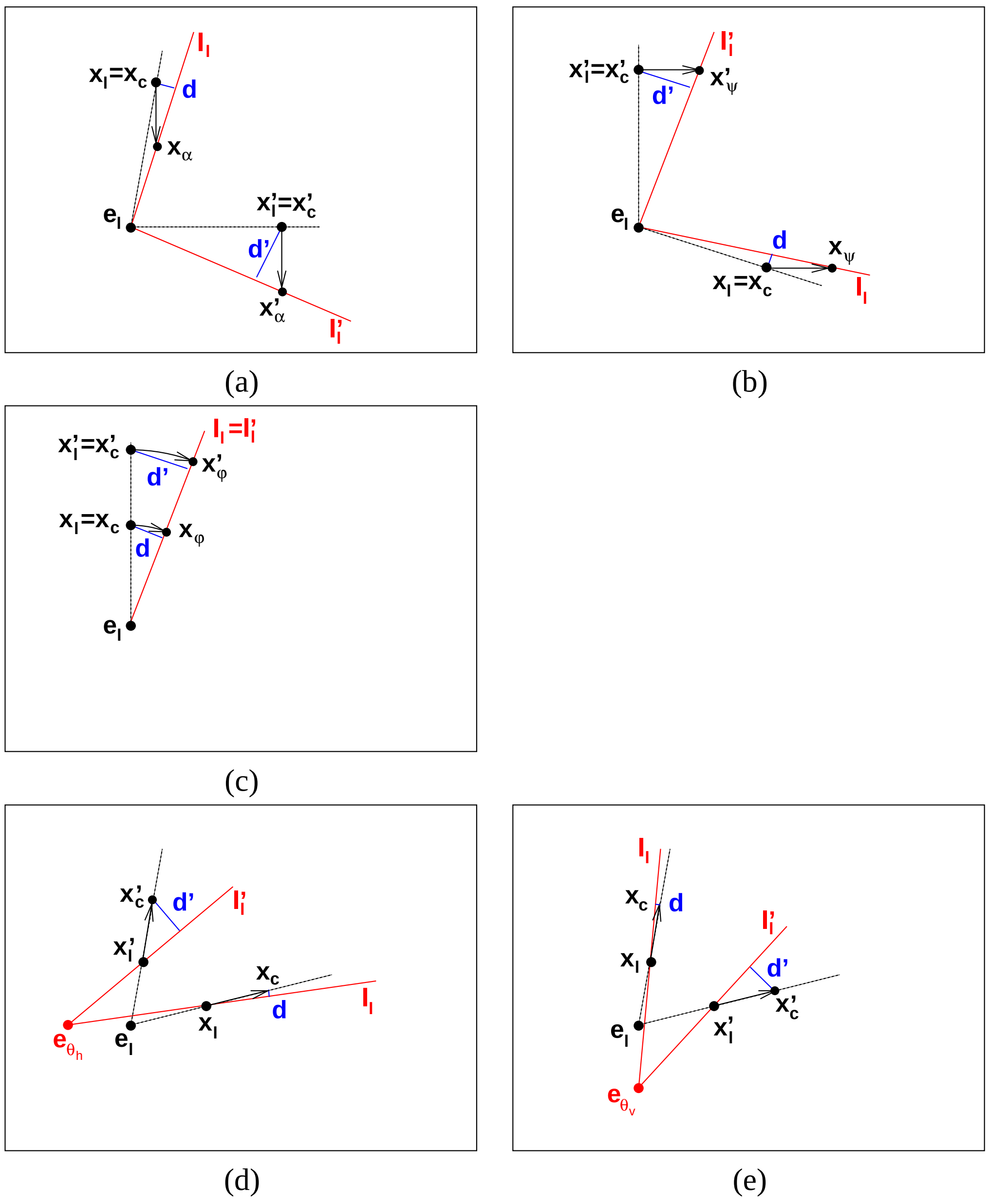
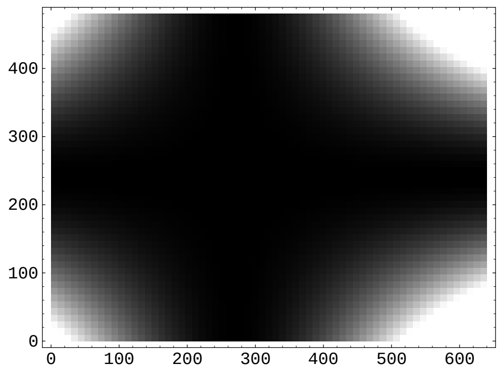
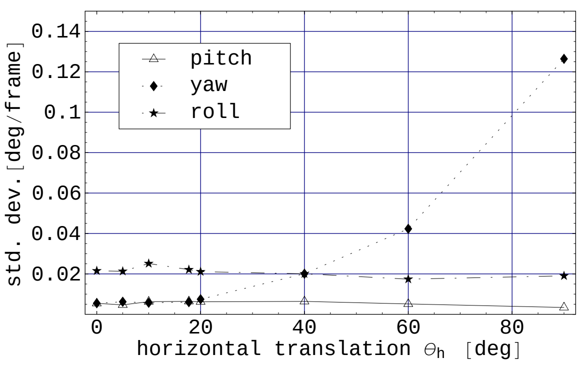


Figure 4.23: Epipolar distances (d and d') depend on the position $\mathbf{x_c}$ and $\mathbf{x_c'}$. For detailed explanation see the text. (a) pitch rate. (b) yaw rate. (c) roll rate. (d) horizontal direction. (e) vertical direction.



(a)



(b)

Figure 4.24: Sideward motion of the camera. (a) Sensitivity of the yaw rate when $\theta_h = 90^\circ$. Dark regions are regions with low sensitivity. White regions have the highest sensitivity of $s_{\Delta\psi,max} = 3.15 * 10^4$. (b) Accuracy of the rotational parameters for increasing values of θ_h.

4.10 Experimental Results

In order to investigate the accuracy of the ego-motion estimation not only simulated data can be used but also real images. This is advantageous since the entire processing from the image aquisition up to the ego-motion estimation is considered. The drawback is that getting ground truth data is cumbersome. A highly accurate IMU could deliver the ground truth. Such a device, however, was not available to the author. Instead standard ESP (Electronic Stability Program) sensors were utilized. The comparison to them is discussed in the next section. An examination of the ego-motion results purely based on the images is discussed in section 4.10.2.

4.10.1 Comparison to Inertial Sensors

Modern vehicles are equipped with ESP, a system engineered for improved vehicle stability. Primarily they are used during severe cornering and on low-friction road surfaces, by helping to reduce over-steering and under-steering. The system intervenes by providing braking forces to the appropriate wheels to correct the path of the vehicle. A yaw rate sensor (beside some others) is required to implement this functionality. When three such sensors are orthogonally aligned we not only measure the yaw rate but also the pitch and the roll rate.

A vehicle equipped with three standard ESP sensors and a camera is used for the data aquisition. An image of an inner-city sequence taken by this vehicle is shown in figure 4.25 along with the optical flow vectors selected for the ego-motion estimation. The ego-motion estimate is

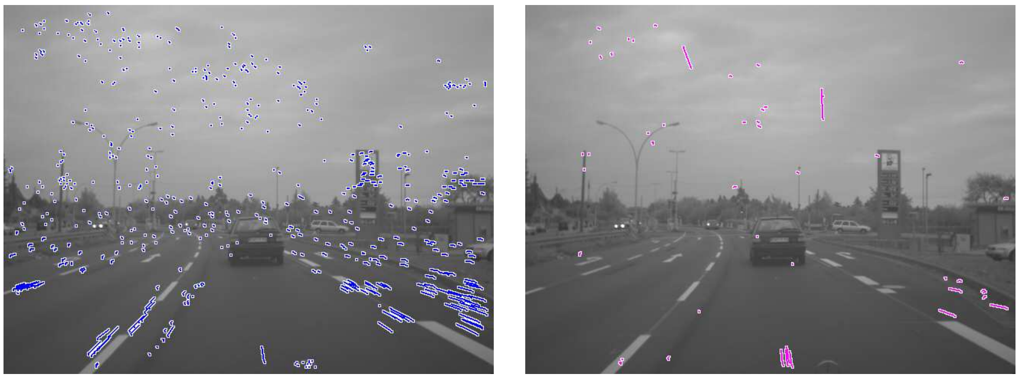
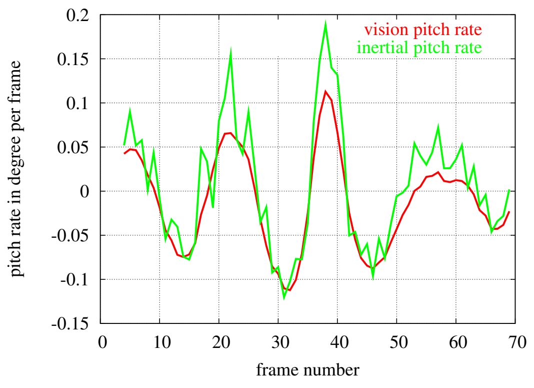

(a) (b)

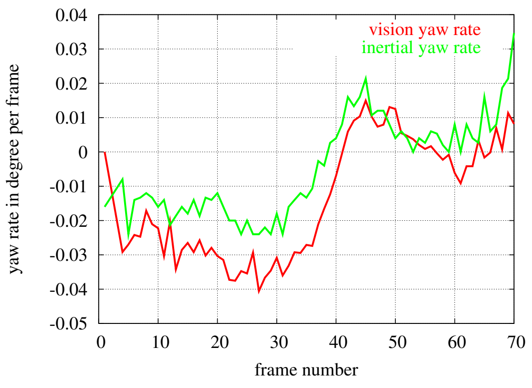
Figure 4.25: An image of an inner-city sequence along with the optical flow vectors selected for the ego-motion estimation. (a) Inliers with $J_{SED} < (1.7\text{px})^2$. (b) Outliers with $J_{SED} \geq (1.7\text{px})^2$.

given by equation 4.17 incorporating the motion model (section 4.6). The threshold T involved in the Huber function which itself is involved in equation 4.17 is set to $T = 1.7$px. It separates the inliers (fig. 4.25a) from the outliers (fig. 4.25b). The obvious mismatched correspondences (long flow vectors in the sky) and the correspondences on IMO's (flow vectors on the (moving) Mercedes star) are correctly detected as outliers.

The estimation result for the entire sequence is shown in figure 4.26. It turns out that in the



(a)



(b)

Figure 4.26: Comparison of the estimated ego-motion to inertial sensors. (a) Pitch rate $\Delta\alpha$. (b) Yaw rate $\Delta\psi$.

case of the pitch rate the data obtained by vision is much smoother compared to that obtained by the inertial sensor. Therefore one can freely assert that the vision based ego-motion is less noisy.

4.10.2 Visual Inspection

The visual inspection method exploits the fact that imaged 3D points located far away are considered invariant against a translation of the camera. Examples for such points are clouds or objects in the background.

The investigation is carried out as follows. The ego-motion of two consecutive frames is estimated. Based on that ego-motion the second image is warped in such a way that the camera rotation vanishes, i.e. the image is stabilized. This is done by applying the infinite homography: $\mathbf{H}_\infty = \mathbf{KRK}^{-1}$. A (virtual) pure translational camera motion remains between the two frames. By using visual inspection, it is examined whether points in the distance have identical positions in both images. Furthermore, the obtained ego-motion is integrated over several frames. In addition this reveals small errors in the ego-motion estimation. Only if the estimates are accurate objects in the distance stay at fixed positions over several frames.

An image sequence recorded out of a truck is used to perform the visual inspection. Figure 4.27a shows one frame of this sequence. The outlined objects in the background (the house and the bridge) are depicted in figure 4.27b. Figure 4.27c shows the outlined image region 30 frames later. During this time period of 1.2 seconds the camera undergoes considerable rotations. Figure 4.27d and 4.27e show the rotation compensated images. The image positions of the house and the bridge in the two images differ about 3 pixels, which is equal to an offset of 0.007 degree per frame.

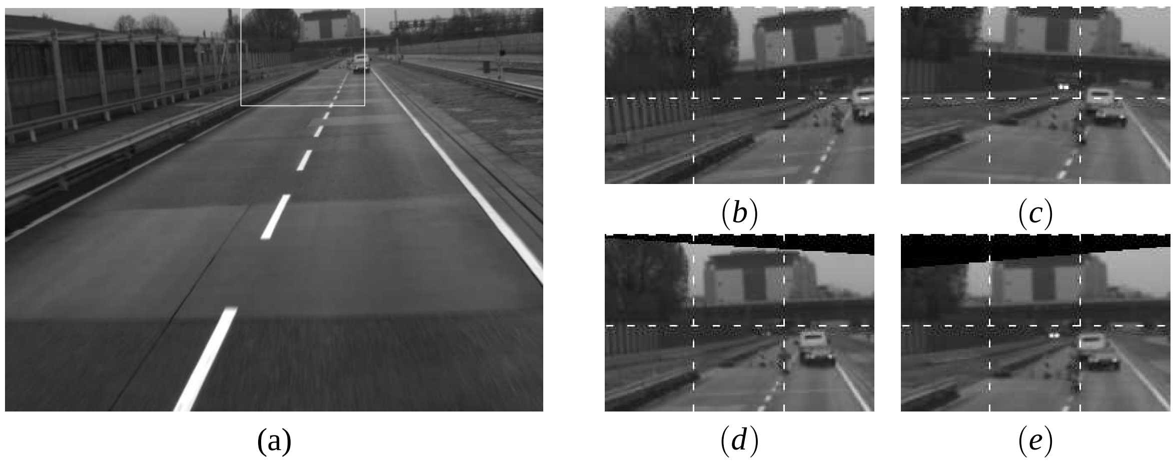


Figure 4.27: Visual inspection of the ego-motion estimation results. (a) Frame 183 of the truck sequence. The installation height of the camera is 2.5*m*, the speed is 50km/h. The region marked white is enlarged in (b). Figure (c) shows the same image region 30 frames later. The camera undergoes considerable rotations. (d) and (e) show the rotation compensated images. The image positions of the house and the bridge differ about 3 pixels.

Chapter 5

Road Homography Estimation

The ego-motion gives us the relative orientation of the camera. This information is not sufficient for a fully featured object detection. We have to know the absolute orientation, too, which is comprised of the absolute angles and the height of the road w.r.t. the camera. The estimation of the road homography gives us this information, except the yaw angle. The estimation relies on the measured optical flow on the road. Only the part of the image containing the road should be considered. In section 5.1, this part, called the *driving corridor*, is computed.

Two types of error metrics are discussed in section 5.2. The recommended geometric error metric is non-linear. Its efficient minimization is addressed in section 5.3. The achievable accuracy of the estimate is investigated in section 5.4. As in the ego-motion case the image regions contribute differently to the estimate. This is shown in section 5.5. The novel Kalman filtering of the road homography is introduced in section 5.6.

5.1 Computation of the Driving Corridor

Using the ego-motion information retrieved by the estimation, the driving corridor is deployed by the extrapolation of the ego-motion. This assumes that the ego-motion is constant over time. Figure 5.1 shows an example of the driving corridor. The lines in the figure show where the camera will be in 1, 2, 3, ... frames.

The marginal points of the driving corridor are now computed. At first the width of the corridor has to be defined. Experiments have shown that in most cases the driver keeps a distance of 0.5m to the roadside. Assuming a vehicle width of 2m the driving corridor sums up to 3m. Inside the driving corridor we expect to see nothing else than the road.

The beginning of the driving corridor is deployed by the world points on the road which are seen by the camera and which are lying as close as possible to the camera. These points are found by mapping the lower image corners to the road using the inverse projection matrix. The projection matrix $\mathbf{P_c}$ maps a world point $\mathbf{x_w} \in \mathbb{P}^3$ to the image poiincludent $\mathbf{x_c}$ in the current frame:

$$\mathbf{x_c} = \mathbf{P_c x_w} \tag{5.1}$$

If the world coordinate frame is chosen such that the road plane coincides with the X-Z plane,

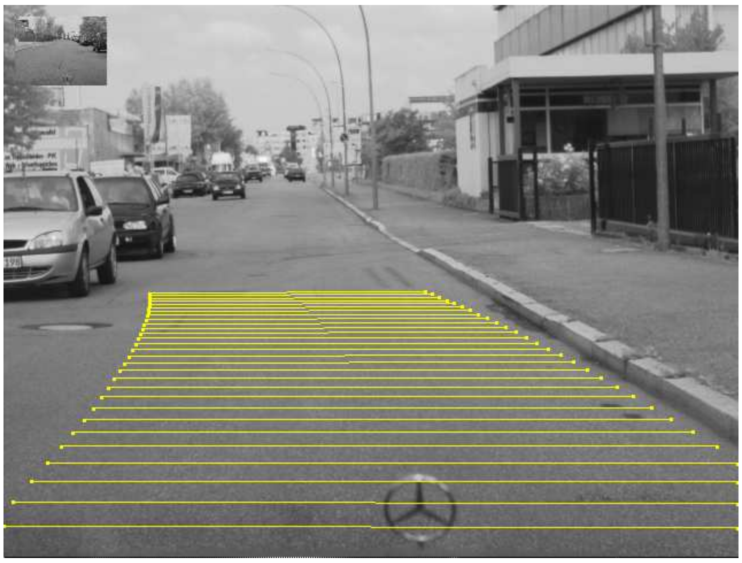


Figure 5.1: Marginal points of the driving corridor. The points of the same time instant are joined by a line. Each line, starting at the lowest and counting upwards, predicts the projected ego-motion one more frame to the future. The ego-motion is predicted 30 frames, which are 1.2 seconds. The width of the corridor is 3m.

the Y-coordinate of a point on the road is zero and equation 5.1 becomes:

$$\mathbf{x_c} = \mathbf{P_c} \begin{pmatrix} X \\ 0 \\ Z \\ W \end{pmatrix} \tag{5.2}$$

The second column of $\mathbf{P_c}$ vanishes and we get a 3x3 invertable matrix $\mathbf{P_c'}$. The point on the road is then computed as follows:

$$\begin{pmatrix} X \\ Z \\ W \end{pmatrix} = \mathbf{P_c'}^{-1}\mathbf{x_c} \tag{5.3}$$

Mapping now the lower image corners (assuming VGA resolution) onto the road yields:

$$\mathbf{x_{w1}} = \mathbf{P_c'}^{-1} \begin{pmatrix} 0 \\ 479 \\ 1 \end{pmatrix} \quad \mathbf{x_{w2}} = \mathbf{P_c'}^{-1} \begin{pmatrix} 639 \\ 479 \\ 1 \end{pmatrix} \tag{5.4}$$

Using the larger depth of these two points $Z_{max} = \max\left(\frac{(\mathbf{x_{w1}})_2}{(\mathbf{x_{w1}})_3}, \frac{(\mathbf{x_{w2}})_2}{(\mathbf{x_{w2}})_3}\right)$ we define two points on the road as the beginning of the driving corridor:

$$\mathbf{x_{cor1}}^{(0)} = \begin{pmatrix} -w/2 \\ 0 \\ Z_{max} \\ 1 \end{pmatrix} \quad \mathbf{x_{cor2}}^{(0)} = \begin{pmatrix} w/2 \\ 0 \\ Z_{max} \\ 1 \end{pmatrix} \tag{5.5}$$

where w is the width of the corridor. The entire corridor is deployed by predicting the vehicle's motion based on the estimated ego-motion. Pitch and roll motions are changing rapidly thus they are hard to predict. However we know they are zero on average. Setting them to zero is a good prediction. The yaw motion changes slowly and is predictable. We just say the current yaw rate will be the same in near future. The term yaw rate here measures the rotation of the vehicle's longitudinal axis within the road plane. It is different from the yaw rate $\Delta\psi$ we got from the ego-motion estimation, due to the rotation sequence $\mathbf{R} = \mathbf{R}(\Delta\alpha, 0, 0) \cdot \mathbf{R}(0, \Delta\psi, 0) \cdot \mathbf{R}(0, 0, \Delta\varphi)$ we used internally in the ego-motion estimation. In order to get the desired yaw rate $\Delta\psi_p$ the Z-axis of the camera is projected onto the road plane (= XZ-plane). This is depicted in figure 5.2. The angle between the last projected Z-axis and the current projected Z-axis gives $\Delta\psi_p$:

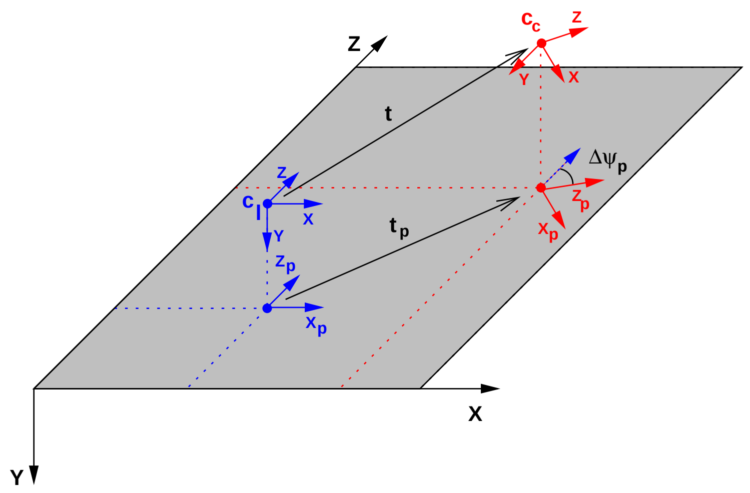


Figure 5.2: Projection of the camera's Z-axis onto the road plane (=XZ-plane). The camera moves from c_l to c_c. The Z-axes are projected onto the plane yielding Z_p. The angle between the last projected Z-axis and the current projected Z-axis is $\Delta\psi_p$.

$$\mathbf{z_{pl}} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \qquad \mathbf{z_{pc}} = \left[\mathbf{R}^T \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right]_{(\cdot)_2=0} = \begin{pmatrix} (\mathbf{R})_{31} \\ 0 \\ (\mathbf{R})_{33} \end{pmatrix} \tag{5.6}$$

$$\Delta\psi_p = \cos^{-1} \frac{\mathbf{z_{pl}}^T \, \mathbf{z_{pc}}}{\|\mathbf{z_{pc}}\|} \tag{5.7}$$

The translation of the camera is also projected onto the road plane. This is done by setting the second component to zero:

$$\mathbf{t_p} = [\mathbf{t}]_{(\cdot)_2=0} \tag{5.8}$$

The camera motion projected onto the road plane ($\mathbf{R_p} = \mathbf{R}(0, \Delta\psi_p, 0)$, $\mathbf{t_p}$) is now used to move the marginal points $\mathbf{x_{cor1}}^{(0)}$ and $\mathbf{x_{cor2}}^{(0)}$:

$$\mathbf{x_{cor1}}^{(n)} = \begin{pmatrix} & \mathbf{R_p} & & \mathbf{t_p} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \mathbf{x_{cor1}}^{(n-1)} \quad \mathbf{x_{cor2}}^{(n)} = \begin{pmatrix} & \mathbf{R_p} & & \mathbf{t_p} \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \mathbf{x_{cor2}}^{(n-1)} \tag{5.9}$$

These recursively formulated equations apply the projected camera motion n-times. Thus the marginal points $\mathbf{x_{cor1}}^{(n)}$ and $\mathbf{x_{cor2}}^{(n)}$ are the predictions n-frames in the future. Finally all these points are projected onto the current frame using the projection matrix $\mathbf{P_c}$. The closed polygon joining the projected points defines the driving corridor. Let Ω denote the set of all image points inside the driving corridor.

5.2 Error Metric

In section 5.1 the driving corridor was computed. It is assumed that there is only the road inside this corridor. The optical flow measured inside the corridor serves as input data for the road homography estimation. As in the case of ego-motion estimation there are two types of error metrics which can be used for the estimation, algebraic and geometric. The algebraic error metric constitutes a closed form solution but is more vulnerable to noise as we will see in the next section. Furthermore it is difficult to integrate *a priori* knowledge. The geometric error metric considers the distances of measured image points to their true (expected) image points.

In general, image points corresponding to a plane in the world are mapped between two images via homography. A general homography has eight DoF. Here we search for a homography which is compatible with the estimated ego-motion $\mathbf{R},\mathbf{t}$. This reduces the DoF to three, namely the pitch angle, the roll angle, and the height. With the internal calibration of the camera $\mathbf{K}$ the (road) homography is composed according to [Hartley & Zisserman 03]:

$$\mathbf{H_r} = \mathbf{K}(\mathbf{R} - \mathbf{e_c}\mathbf{v}^T)\mathbf{K}^{-1} \tag{5.10}$$

where $\mathbf{e_c} = -\mathbf{Rt}$ is the current epipole expressed in normalized coordinates. The three dimensional vector $\mathbf{v}$ encodes the plane normal $\mathbf{n} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$ and the distance (height) of the camera to the plane: $h = \frac{1}{\|\mathbf{v}\|}$. The plane normal itself is defined by the pitch angle α and the roll angle φ of the road w.r.t. the camera:

$$\mathbf{n} = \mathbf{R}(\alpha, 0, \varphi)(0, -1, 0)^T \tag{5.11}$$

With the homography $\mathbf{H_r}$, an image point $\mathbf{x_l}$ in the last frame is mapped to the corresponding point $\mathbf{x_c}$ in the current frame according to:

$$\mathbf{x_c} = \mathbf{H_r}\mathbf{x_l} \tag{5.12}$$

In the next section the homography estimation method considering the algebraic error metric is described. The geometric counterpart which is recommended is described in section 5.2.2.

5.2.1 Algebraic Error Metric

The algebraic error metric which is presented now stems from [Hartley & Zisserman 03]. The aim is to solve linearly for the $\mathbf{v}$ vector given a set of correspondences. To this end we first eliminate the $\mathbf{K}$ matrix by a normalization of the image coordinates: $\mathbf{x}'_\mathbf{l} = \mathbf{K}^{-1}\mathbf{x_l}$ and $\mathbf{x}'_\mathbf{c} = \mathbf{K}^{-1}\mathbf{x_c}$. Each correspondence generates a linear constraint on $\mathbf{v}$ as

$$\mathbf{x}'_\mathbf{c} = (\mathbf{R} - \mathbf{e_c}\mathbf{v}^T)\mathbf{x}'_\mathbf{l} = \mathbf{R}\mathbf{x}'_\mathbf{l} - \mathbf{e_c}(\mathbf{v}^T\mathbf{x}'_\mathbf{l}) \tag{5.13}$$

However equation 5.13 cannot be used directly, since $\mathbf{x'_c}$ is a homogeneous point with the third component fixed by the right-hand side. We are only able to measure inhomogeneous image points. Homogeneous points represent the same inhomogeneous points iff they are parallel (= cross product is zero). When taking the cross product:

$$\mathbf{x'_c} \times [\mathbf{Rx'_l} - \mathbf{e_c}(\mathbf{v}^T\mathbf{x'_l})] = (\mathbf{x'_c} \times \mathbf{Rx'_l}) - (\mathbf{x'_c} \times \mathbf{e_c})(\mathbf{v}^T\mathbf{x'_l}) = 0 \tag{5.14}$$

we can put in our inhomogeneous measurements into $\mathbf{x'_c}$ and $\mathbf{x'_l}$. Forming the scalar product with the vector $(\mathbf{x'_c} \times \mathbf{e_c})$ gives:

$$\mathbf{x'_l}^T\mathbf{v} = \frac{(\mathbf{x'_c} \times \mathbf{Rx'_l})^T(\mathbf{x'_c} \times \mathbf{e_c})}{(\mathbf{x'_c} \times \mathbf{e_c})^T(\mathbf{x'_c} \times \mathbf{e_c})} = b \tag{5.15}$$

Each correspondence generates an instance of the equation above. Having several correspondences one stacks all equations together yielding the linear equation system: $\mathbf{Mv} = \mathbf{b}$. The rows of the matrix $\mathbf{M}$ are the single points $\mathbf{x'_c}$. $\mathbf{v}$ is found by the least squares solution:

$$\mathbf{v} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{b} \tag{5.16}$$

This method works well if no outliers are present in the optical flow. However, in real life we expect outliers. In order to be robust the method is augmented with the iteratively reweighted least squares (IRLS) approach. It incorporates the concept of M-estimation that we already applied to the ego-motion estimation. The higher the residual of a correspondence, the lower its weight. Thus the influence of "bad" correspondences is attenuated. The residual vector is given by $\delta = \mathbf{Mv} - \mathbf{b}$. We apply the Huber cost function $C(\delta)$ again to compute the single weights:

$$(\mathbf{w})_i = \frac{\sqrt{C((\delta)_i)}}{|(\delta)_i|} \tag{5.17}$$

The linear equation system $\mathbf{Mv} = \mathbf{b}$ is extended by the diagonal weight matrix $\mathbf{W} = diag((\mathbf{w})_1, (\mathbf{w})_2, \ldots)$ and then solved for $\mathbf{v}$:

$$\mathbf{WMv} = \mathbf{Wb} \tag{5.18}$$

Defining the weights, as in equation 5.17, the least squares solution of 5.18 effectively minimizes the sum of the Huber evaluated residuals $\sum_i C((\delta)_i)$. The equation system 5.18 is solved multiple times until convergence. Each time the weights are updated using the solution of $\mathbf{v}$ from the last time. The weights are initially set to 1.

5.2.2 Geometric Error Metric

The geometric error metric represent a geometric meaningful residual, namely the parallax vector μ:

$$\mu = \begin{pmatrix} (\mathbf{x_c})_1 - \frac{(\mathbf{H_r x_l})_1}{(\mathbf{H_r x_l})_3} \\ (\mathbf{x_c})_2 - \frac{(\mathbf{H_r x_l})_2}{(\mathbf{H_r x_l})_3} \end{pmatrix} \tag{5.19}$$

The $\mathbf{v}$ vector capturing the road normal and the camera height is found by minimization of the sum of all parallax vectors:

$$\hat{\mathbf{v}} = \arg\min_{\mathbf{v}} \sum_{i=1}^{N_n} C_p((\mu(\mathbf{v}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_1)^2 + C_p((\mu(\mathbf{v}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_2)^2 \quad \text{with} \quad \mathbf{x}_{\mathbf{l},i} \in \Omega \tag{5.20}$$

with C_p being the point-symmetric rooted Huber cost function introduced in the ego-motion chapter (eq. 4.34, p. 61). We apply Levenberg-Marquardt (LM) to perform the minimization. The terms $C_p((\mu(\mathbf{v}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_1)$ and $C_p((\mu(\mathbf{v}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_2)$ serve as residuals. In section 5.3 we will see that the assumptions made by the LM algorithm are not fulfilled exactly but nevertheless LM performs well.

The iterative minimization allows easily the incorporation of *a priori* knowledge, such as the height h of the camera. Doing this reduces the DoF to two, pitch angle α and roll angle φ. However, we do not minimize over α and φ directly but over the first and the third component of the normal vector, i.e. $(\mathbf{n})_1$ and $(\mathbf{n})_3$ respectively. The second component is enforced such that $\|\mathbf{n}\| = 1$: $(\mathbf{n})_2 = -\sqrt{1 - (\mathbf{n})_1^2 - (\mathbf{n})_3^2}$. This parameterization circumvents trigonometric functions.

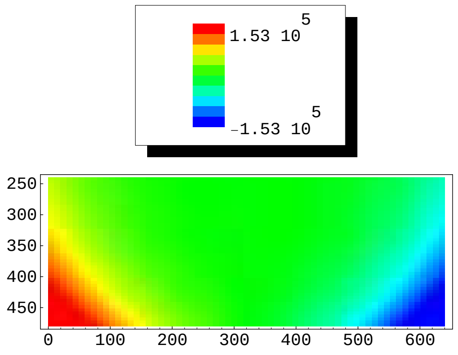
5.3 Efficient Minimization

The geometric error metric (eq. 5.20) is non-linear and hence must be minimized iteratively. When we dealt with the ego-motion estimation, we already discussed minimization schemes and found out that the Levenberg-Marquardt (LM) method is very efficient. The LM minimization works well only if the assumptions made by this algorithm are fulfilled. Fulfilled assumptions legitimate to shorten the computation of the Hessian matrix. We recapitulate equation 4.32 on page 61:

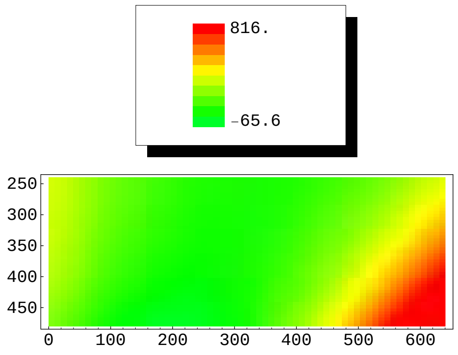
$$\frac{\partial^2 \chi}{\partial(\mathbf{p})_k \partial(\mathbf{p})_l} = 2\sum_i \frac{\partial r_i}{\partial(\mathbf{p})_l} \frac{\partial r_i}{\partial(\mathbf{p})_k} + r_i \frac{\partial^2 r_i}{\partial(\mathbf{p})_k \partial(\mathbf{p})_l} \tag{5.21}$$

LM assumes that the term $r_i \frac{\partial^2 r_i}{\partial(\mathbf{p})_k \partial(\mathbf{p})_l}$ is small compared to $\frac{\partial r_i}{\partial(\mathbf{p})_l} \frac{\partial r_i}{\partial(\mathbf{p})_k}$ and thus is neglected. In the minimum the residuals are zero-mean, and the assumption is fulfilled. However, the task is to find the minimum given a close starting point. The zero-mean property of the residuals does not hold outside the minimum. It follows that the second derivatives $\frac{\partial^2 r_i}{\partial(\mathbf{p})_k \partial(\mathbf{p})_l}$ itself have to be small compared to the product of the first derivatives. But this is not the case for the derivatives containing the roll angle φ. Figure 5.3 shows examples of $\frac{\partial r_i}{\partial \alpha} \frac{\partial r_i}{\partial \varphi}$ and $\frac{\partial^2 r_i}{\partial \alpha \partial \varphi}$. Thereby, the camera moves rotation free along the optical axis, i.e. $\mathbf{p_e} = (0,0,0,0,0)^T$. The optical axis is parallel to the road and the camera height is $1m$, i.e. $\mathbf{v} = (0,-1,0)^T$. The derivatives depend on the image position. Only in some regions the assumption is fulfilled.

The error function χ incorporates a set of (equally distributed) correspondences, i.e. the actual question is $\sum_i \frac{\partial r_i}{\partial \alpha} \frac{\partial r_i}{\partial \varphi} \gg \sum_i \frac{\partial^2 r_i}{\partial \alpha \partial \varphi}$? A good approximation of the sum is to consider the



(a)



(b)

Figure 5.3: The first and the second derivatives of the residual $r = (\mu)_1 + (\mu)_2$ w.r.t. the pitch angle α and roll angle φ. The derivatives depend on the image position. (a) $\frac{\partial r}{\partial \alpha}\frac{\partial r}{\partial \varphi}$. (b) $\frac{\partial^2 r}{\partial \alpha \partial \varphi}$.

integral over the entire lower image half:

$$\int_{240}^{480}\int_{0}^{640} \left.\frac{\partial r_i}{\partial \alpha}\frac{\partial r_i}{\partial \varphi}\right|_{\substack{\mathbf{p_e}=(0,0,0,0,0)^T \\ \mathbf{v}=(0,-1,0)^T}} du dv = 0 \tag{5.22}$$

$$\int_{240}^{480}\int_{0}^{640} \left.\frac{\partial^2 r_i}{\partial \alpha \partial \varphi}\right|_{\substack{\mathbf{p_e}=(0,0,0,0,0)^T \\ \mathbf{v}=(0,-1,0)^T}} du dv = 1.7\cdot 10^7 \tag{5.23}$$

Oops! The assumption is not fulfilled. A similar result is obtained when $\frac{\partial r_i}{\partial \varphi}\frac{\partial r_i}{\partial h}$ is considered. By the way, all other derivatives behave inconspicuously.

Does this violation spoil the minimization speed? Not really. The LM method is still very efficient. On average, 4.2 calls of the function plus derivatives are enough to find the minimum. The entire minimization is performed in 0.7 ms on a Pentium IV 2.4 GHz when 300 correspondences are utilized.

5.4 Accuracy of Road Homography Estimation

In this section, we investigate the accuracy of the road homography estimation based on synthetic data. We compare the algebraic to the geometric error metric, and to the geometric metric with given camera height. It is not necessary to estimate the height, since it can be determined by a calibration.

The synthetic data consists of 100 world points lying on the road. They are equally distributed in the last image frame. The points are mapped into the current frame using the ground-truth road homography. There the points get an additive noise according to $N(0,\sigma)$. Some of the points are not mapped by the homography. Instead, their position in the current frame is the position in the last frame plus an additive noise according to $N(0,30\text{px})$. These points represent outliers.

The road homography is estimated 100 times while varying the world points and the ground-truth road homography each time. Concretely, the pitch and the roll angle of the road are uniformly distributed in the range $-20..20°$. The height of the camera is within $1..2m$. The ego-motion is constant: the camera moves $1m$ along the optical axis and does not rotate. In all simulations the focal length of the camera is $f_x = f_y = 1000$px.

The mean of the estimation error is nearly zero in all simulations. It seems that both error metrics produce unbiased estimates. More interesting than the mean is the standard deviation of the estimation error, shown in figure 5.4. The geometric metric produces more accurate results than the algebraic metric. Especially the accuracy of the pitch angle is better. Using the camera height as *a priori* knowledge (figure 5.5) increases the pitch angle's accuracy considerably (factor of 3). This is not surprising since certain combinations of camera height and pitch angle produce a similar flow field for the road, i.e. these two entities are correlated. An example is depicted in figure 5.6.

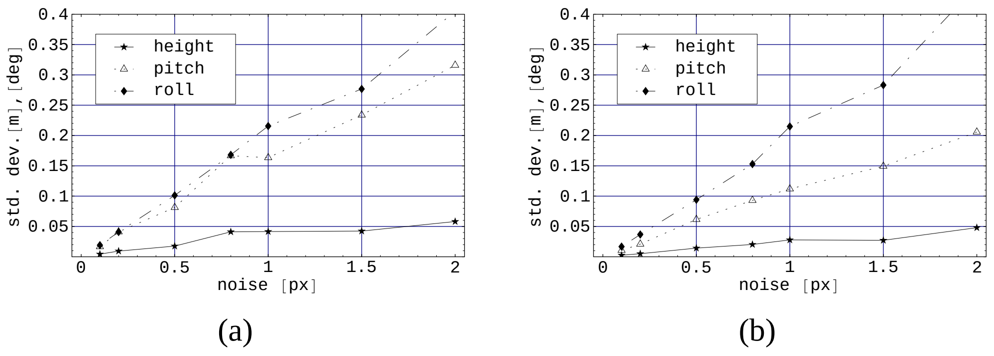


Figure 5.4: Accuracy of the road homography estimation in dependence of the noise level. (a) algebraic metric (equation 5.18). (b) geometric metric (equation 5.20).

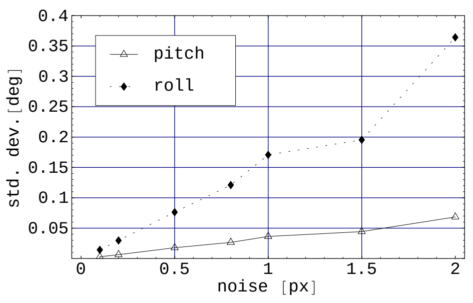
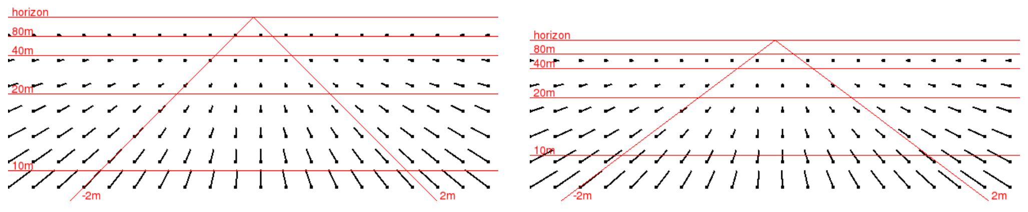


Figure 5.5: Accuracy of the road homography estimation in dependence of the noise level. The geometric metric is minimized using the camera height as *a priori* knowledge.

Beside the noise influence we investigate the robustness of the road homography estimation. Figure 5.7 shows how the accuracy evolves when the outlier fraction increases. Up to 60% outliers the estimate is influenced hardly. Higher outlier fractions degrade the estimate drastically.



(a) (b)

Figure 5.6: Different combinations of camera height h and pitch angle α produce a similar flow field for the road. (a) $h = 2\text{m}$, $\alpha = 0°$. (b) $h = 1.5\text{m}$, $\alpha = -1.7°$

In this simulation the noise level is set to $\sigma = 0.55\text{px}$ and the geometric metric is used. The algebraic metric behaves similar to the geometric one.

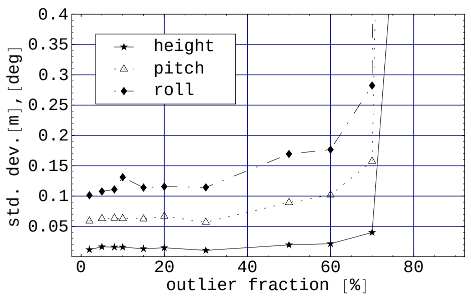


Figure 5.7: Accuracy of the road homography estimation depending on the outlier fraction. The geometric metric is minimized. The estimation is robust up to 60% outliers.

We have seen that the road homography estimation is much more accurate if the camera height is not estimated but given. Thereby we assumed the height is error free, which does not hold in practice. Therefore, we ask: To which extent does an uncertain height spoil the estimate? When is it better to forbear from the given height and estimate it, instead? To answer these questions we carry out another simulation. This time the noise in the height is varied. The noise in the optical flow is constant with $\sigma = 0.55\text{px}$. There are no outliers. Figure 5.8 shows the resulting accuracy. For an uncertainty (standard deviation) of 0.015m the pitch angle's standard deviation is 0.06°. This value is also obtained if the height is estimated (compare to fig. 5.4b), i.e. up to an uncertainty of 0.015m it makes sense to trust the given height.

Until now, we have discussed the effect of an uncertain camera height. What is if the driven distance d between the frames (retrieved by odometry) is uncertain? The answer is: The effect is nearly the same. To see this the homography matrix is composed using vectors of length one:

$$\mathbf{H_r} = \mathbf{K}(\mathbf{R} + \frac{d}{h}\mathbf{R}\mathbf{t_{l1}}\mathbf{n}^T)\mathbf{K}^{-1} \tag{5.24}$$

The camera translation **t** is split into $d = ||\mathbf{t}||$ and $\mathbf{t_{l1}} = \frac{\mathbf{t}}{d}$. The homography depends only on the relation $\frac{d}{h}$. This means that small uncertainties in h have the same effect as small uncertainties in d. Also we can estimate the relation $\frac{d}{h}$ rather than the height h. In other words, if the height h is given, for example via offline calibration, we can estimate the driven distance d which makes us independent from the odometry.

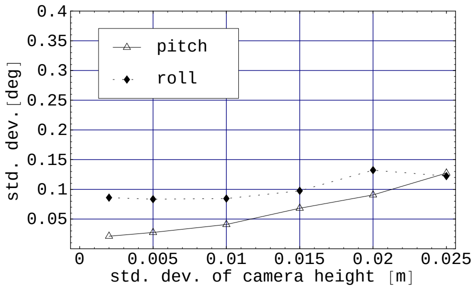


Figure 5.8: Accuracy of the road homography estimation in case of an uncertain camera height.

5.5 Sensitivity of the Road Homography Parameters

The estimation of the road homography is performed by minimization of the geometric error metric (eq. 5.20). The accuracy of the estimated parameters depends on the curvature (2nd partial derivatives) of the error metric in the minimum $\hat{\mathbf{v}}$. The higher the curvature, the more sensitive the parameter. In the following we investigate the sensitivities of the parameters pitch angle α, roll angle φ, and camera height h. We prefer these parameters to **v**, due to their better physical representation. They are directly related to **v** and thus do not effect the estimation result. Their sensitivities are given by:

$$s_\alpha = \left. \frac{\partial^2 \left[(\mu)_1^2 + (\mu)_2^2\right]}{\partial\alpha^2} \right|_{\mathbf{v}=\hat{\mathbf{v}}} \tag{5.25}$$

$$s_\varphi = \left. \frac{\partial^2 \left[(\mu)_1^2 + (\mu)_2^2\right]}{\partial\varphi^2} \right|_{\mathbf{v}=\hat{\mathbf{v}}} \tag{5.26}$$

$$s_h = \left. \frac{\partial^2 \left[(\mu)_1^2 + (\mu)_2^2\right]}{\partial h^2} \right|_{\mathbf{v}=\hat{\mathbf{v}}} \tag{5.27}$$

As for the ego-motion case, the parameters have different sensitivities. Furthermore, the sensitivity depend on the image position. Figure 5.9 shows this for the standard ego-motion case (translation along the optical axis, no rotation) and standard road homography case ($\alpha = 0°$, $\varphi = 0°$, and $h = 1m$). All three parameters have their maximum sensitivity at the lower left and lower right image corner. In theory, taking correspondences out of these regions lead to the most accurate estimate. However, in these regions the motion blur also reaches its maximum value causing more noise in the correspondences. In practice we choose an equally distributed subset from the available correspondences.

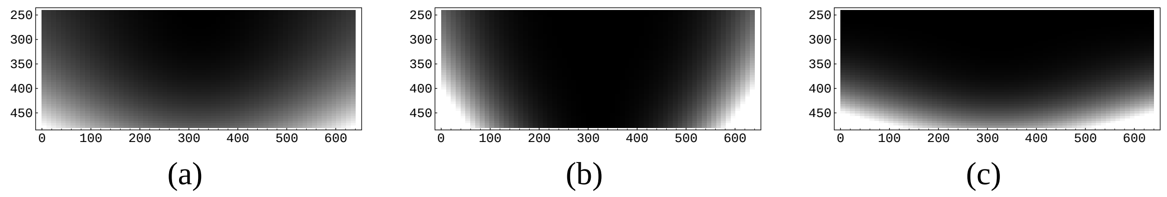


Figure 5.9: Sensitivities of the road homography parameters depending on the image position. Dark regions are regions with low sensitivity. White regions have the maximum sensitivity for that parameter. (a) pitch angle. $s_{\alpha,\max} = 9.59 * 10^5$ (b) roll angle. $s_{\varphi,\max} = 9.82 * 10^4$ (c) height. $s_{h,\max} = 5.52 * 10^4$

5.6 Road Homography Filtering

The estimation of the road homography relies on the availability of a well textured road. However, sometimes the road is low-textured or even homogeneously textured. In that case the homography estimation performs poorly and measurements of the absolute pitch and roll angles fail. Only the ego-motion estimate is reliable, i.e. we are only able to estimate rotations from frame to frame. Under the assumption that the road has a constant vertical slope, the rotations from frame to frame are identical to the temporal derivatives of the absolute angles. Figure 5.10 illustrates this for the pitch angle α. In times where α is estimated badly, it can be updated through an integration of the pitch rates $\Delta\alpha$. In this section, we develop a Kalman filter based approach, which uses the ego-motion to stabilise the homography estimation in this way. It has been published in [Klappstein *et al.* 07a].

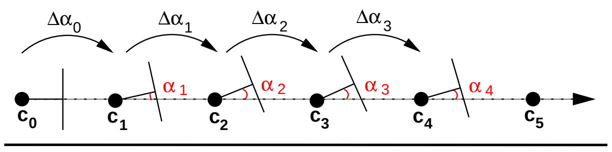


Figure 5.10: The camera moves from c_0 to c_4 while pitching. The absolute pitch angle α_i of the road w.r.t. the camera is the sum of the pitch rates $\Delta\alpha_i$. This assumes a road with constant vertical slope and known initial pitch angle α_0: $\alpha_i = \alpha_0 + \sum_0^i \Delta\alpha_i$

A block diagram of the approach is shown in figure 5.11. At first the ego-motion is estimated utilizing the current pitch angle α as the vertical translational direction θ_v. This feedback we had already built in in section 4.6.3, where we had discussed the motion model of the vehicle. There the estimated pitch rates were only integrated and fed back, which led to unstable behavior. Now, with the additional measurement of the absolute pitch angle, the behavior becomes stable as the experimental results will show.

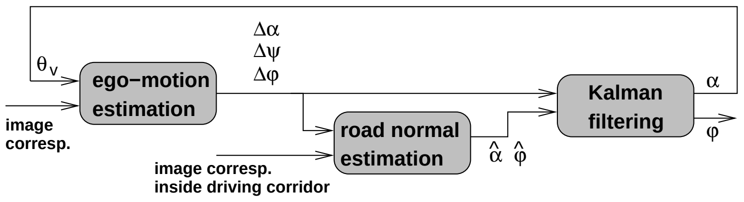


Figure 5.11: Block diagram of the homography filtering approach. At first the ego-motion is estimated utilizing the current pitch angle α as the vertical translational direction θ_v. The result is used to estimate the normal vector of the road. In the last step both results are combined in a Kalman filter.

In the second step, the result of the ego-motion estimation is used to estimate the road homography. Following the outcome of the accuracy investigation, section 5.4, we forbear from the estimation of the camera height and the driven distance. Instead, we assume that the camera height is given and that an accurate odometer is in use. This increases the estimation accuracy of the remaining parameters, the absolute pitch and the absolute roll angle. These two parameters define the normal vector of the road. In the last step the ego-motion estimate and the road normal estimate are combined in a Kalman filter.

The quality of the estimated homography varies with the "texturedness" of the road. A low-textured road makes it hard to establish image correspondences. In such a case, we cannot trust the estimate, so we put our confidence into the ego-motion estimate. To this end the Kalman filter requires a statement about the (un)certainties of the estimated ego-motion and homography. This statement is developed in the following sections.

5.6.1 Uncertainty of an Estimate

In general, an estimate results from uncertain (inaccurate) input data. For example, the estimated road homography results from uncertain image correspondences. Commonly, the uncertainty is expressed with the covariance matrix. If the relation between the estimate and the input data is explicit, a first order approximation of the uncertainty of the estimate is computed by the well-known covariance propagation: Let $\hat{\mathbf{p}}$ be the estimate, $\mathbf{x}$ the input data, and $\mathbf{f}$ an explicit function such that $\hat{\mathbf{p}} = \mathbf{f}(\mathbf{x})$. The covariance matrix $Cov[\hat{\mathbf{p}}]$ of the estimate is then:

$$Cov[\hat{\mathbf{p}}] = \mathbf{J} \cdot Cov[\mathbf{x}] \cdot \mathbf{J}^T \tag{5.28}$$

where $\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\Big|_{\bar{\mathbf{x}}}$ is the Jacobian matrix evaluated at the mean $\bar{\mathbf{x}}$ of $\mathbf{x}$. In practice one evaluates $\mathbf{J}$ at the concrete measured value of $\mathbf{x}$, assuming that the value is sufficiently close to the mean.

However, if the relation is implicit, i.e. $\mathbf{f}(\mathbf{x}, \hat{\mathbf{p}}) = 0$, things become more complicated. This is the case when $\hat{\mathbf{p}}$ is obtained as the minimum of some error function χ, since the gradient $\mathbf{f} = \frac{\partial \chi}{\partial \mathbf{p}}$ has to be zero. Faugeras and Luong [Faugeras & Luong 01] applied the implicit functions theorem to derive the covariance propagation for this case. The theorem says: if the Jacobian $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ is invertable at $\hat{\mathbf{p}}$ the implicit function can be locally transformed into an explicit one $\mathbf{f}_{\text{exp}}$, and its

Jacobian is given by:

$$\frac{\partial \mathbf{f}_{\text{exp}}}{\partial \mathbf{x}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{p}}\right)^{-1} \cdot \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \tag{5.29}$$

To get the explicit Jacobian of the implicit error function χ its gradient is put into equation 5.29 yielding:

$$\frac{\partial \mathbf{f}_{\text{exp}}}{\partial \mathbf{x}} = -\left(\frac{\partial^2 \chi}{\partial \mathbf{p}^2}\right)^{-1} \cdot \frac{\partial \chi}{\partial \mathbf{p} \partial \mathbf{x}} \tag{5.30}$$

This Jacobian evaluated at $\hat{\mathbf{p}}$ can be used as $\mathbf{J}$ in the covariance propagation, equation 5.28. Thus we have a first order approximation of the estimate's covariance matrix, even if the relation between the estimate and the input data is implicit.

Faugeras and Luong [Faugeras & Luong 01] adapted the covariance propagation for the case that χ is a least-squares error function: $\chi = \sum_i^N r(\mathbf{p}, \mathbf{x}_i)^2$. Under the following assumptions:

1. The terms $r_i \frac{\partial^2 r_i}{\partial \mathbf{p}^2}$ are negligible with respect to the terms $\left(\frac{\partial r_i}{\partial \mathbf{p}}\right)^T \frac{\partial r_i}{\partial \mathbf{p}}$. The same assumption is made by the Levenberg-Marquardt minimization.
2. The $\mathbf{x}_i$'s are independent.
3. The residuals r_i are independent and identically distributed.
4. The mean of the r_i's at the minimum is zero.

the covariance propagation can be simplified:

$$Cov[\hat{\mathbf{p}}] = \frac{2\chi_{\min}}{N - \dim(\mathbf{p})} \mathbf{H}^{-T} \tag{5.31}$$

with $\chi_{\min} = \sum_{i=1}^N r_i$ and $\mathbf{H} = 2\sum_{i=1}^N \left(\frac{\partial r_i}{\partial \mathbf{p}}\right)^T \frac{\partial r_i}{\partial \mathbf{p}}\Big|_{\mathbf{p}=\hat{\mathbf{p}}}$ the approximate Hessian matrix in the minimum $\hat{\mathbf{p}}$.

5.6.2 Uncertainty of the Ego-Motion Estimate

We now use the general formulation, discussed in the last section, to compute the uncertainty of the ego-motion. To this end we recapitulate the error metric which is minimized:

$$\hat{\mathbf{p}}_\mathbf{e} = \arg\min_{\mathbf{p}_\mathbf{e}} \sum_{i=1}^{N_e} \underbrace{C_p(J_{RSED,i})}_{r_i}{}^2 \tag{5.32}$$

The vector $\mathbf{p}_\mathbf{e} = (\Delta\alpha, \Delta\psi, \Delta\varphi)^T$ contains the ego-motion parameters. The assumptions made in conjunction with equation 5.31 are (nearly) fulfilled. In section 4.7, we had shown that the second derivatives of r_i with respect to $\mathbf{p}_\mathbf{e}$ are negligible (1st assumption). Of course the image correspondences (= $\mathbf{x}_i$'s) are independent (2nd assumption) as well as the residuals r_i's, since every individual correspondence is made from its own image region.

The residuals depend on the image correspondences, which themselves depend on the texture's "cornerness" in the image. The lower the cornerness the more inaccurate the correspondences. Thus, the correspondences as well as the residuals are not identically distributed (3rd assumption). However, the optical flow algorithm used in this thesis excludes image regions of low cornerness, so the correspondences have a comparable accuracy. Furthermore, we model their covariance matrices as a multiple of the identity matrix.

Next, we show that the residuals r_i's are zero-mean at the minimum (4th assumption). In fact, we show that every individual residual is zero-mean, rather than the set of all residuals. To this end, we consider the true ego-motion and the noise in the correspondences. It is assumed that the minimum coincides with the true ego-motion. The noise in the correspondences is modeled as follows. The point $\bar{\mathbf{x}}_\mathbf{l}$ in the last frame is measured error-free. Its corresponding point $\mathbf{x_c}$ in the current frame is measured with an error $\mathbf{n} = ((\mathbf{n})_1, (\mathbf{n})_2, 0)^T$: $\mathbf{x_c} = \bar{\mathbf{x}}_\mathbf{c} + \mathbf{n}$ where $\bar{\mathbf{x}}_\mathbf{c}$ is the (homogenized) true point. $\mathbf{n}$ is modeled as a zero-mean normal distribution: $\mathbf{n} \sim N(0, Cov[\mathbf{n}])$ with the propability distribution function $\mathrm{pdf}(\mathbf{n}) = \frac{1}{(2\pi)^{3/2}|Cov[\mathbf{n}]|^{1/2}} e^{-\frac{1}{2}\mathbf{n}^T Cov[\mathbf{n}]^{-1}\mathbf{n}}$.

Formally, the zero-mean property of the residual $r = r(\mathbf{F}, \bar{\mathbf{x}}_\mathbf{l}, \mathbf{x_c}) = C_p(J_{RSED}(\mathbf{F}, \bar{\mathbf{x}}_\mathbf{l}, \mathbf{x_c}))$ is verified by showing that the following equation holds:

$$\iint_{-\infty}^{\infty} r \cdot \mathrm{pdf}(\mathbf{n})\, d\mathbf{n} = 0 \quad \forall\, \mathbf{F}, \bar{\mathbf{x}}_\mathbf{l} \leftrightarrow \bar{\mathbf{x}}_\mathbf{c} \tag{5.33}$$

The double integral means that the first and the second component of $\mathbf{n}$ is integrated. Instead of solving this unaesthetic integral[1], we will show geometrically that the residual is zero-mean. The rooted symmetric epipolar distance J_{RSED} consists of two parts: the distance of $\bar{\mathbf{x}}_\mathbf{l}$ to its epipolar line $\mathbf{l_l} = \mathbf{F} \cdot \mathbf{x_c}$ measured in the last frame, and the distance of $\mathbf{x_c}$ to its epipolar line $\bar{\mathbf{l}}_\mathbf{c} = \mathbf{F} \cdot \bar{\mathbf{x}}_\mathbf{l}$ measured in the current frame. The noise acts differently on these distances. In the current frame it shifts $\mathbf{x_c}$ around the true point, whereas in the last frame it varies the epipolar line $\mathbf{l_l}$. This results in different statistical behaviors of the two epipolar distances.

Figure 5.12 shows the current frame with its epipolar distance. The uncertain point $\mathbf{x_c}$ is characterized by its covariance ellipse, representing positions of $\mathbf{x_c}$ of constant propability. It can be seen that the epipolar distance d_c is point-symmetric regarding the noise $\mathbf{n}$, i.e.: $d_c(\mathbf{n}) = -d_c(-\mathbf{n})$. Please note that d_c is positive or negative depending on whether $\mathbf{x_c}$ lies on the "left" or on the "right" side of $\bar{\mathbf{l}}_\mathbf{c}$. Although, the terms "left" and "right" are wacky unless a mathematical meaning is given to them, it is hoped that the reader understand what is meant. Any point-symmetric function maintains the zero-mean property if the argument of the function is symmetrically distributed. With $\mathbf{n} \sim N(0, Cov[\mathbf{n}])$, this is the case and d_c is zero-mean.

What about the epipolar distance d_l in the last frame? Is it also zero-mean? Figure 5.13 shows how the noise acts on the epipolar line $\bar{\mathbf{l}}_\mathbf{l}$. For better visualization a rotation-free ego-motion is chosen, producing points $\bar{\mathbf{x}}_\mathbf{c}$ lying on $\bar{\mathbf{l}}_\mathbf{l}$. This comes without loss of generality. Again the uncertainty of $\mathbf{x_c}$ is represented by its covariance ellipse. $\mathbf{x_c}$ together with the epipole $\mathbf{e_c}$ form the epipolar line $\mathbf{l_l}$. A point $\mathbf{x}'_\mathbf{c}$ is defined as the result from reflecting $\mathbf{x_c}$ over the true epipolar line $\bar{\mathbf{l}}_\mathbf{l}$. In the same way the reflected epipolar line $\mathbf{l}'_\mathbf{l}$ is defined. The epipolar distances d_l and d'_l are identical except the sign: $d_l = -d'_l$. If the covariance ellipse is a circle, as in figure 5.13a,

[1]A closed-form solution does not exist.

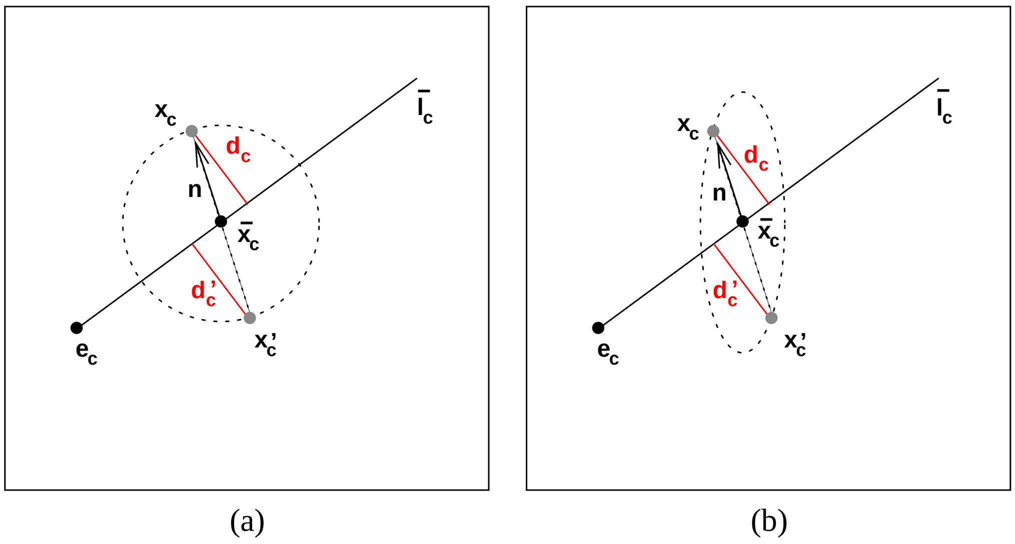


Figure 5.12: The epipolar distance d_c in the current frame is point-symmetric regarding the noise $\mathbf{n}$ in the measured point $\mathbf{x_c} = \bar{\mathbf{x}}_\mathbf{c} + \mathbf{n}$. d_c is the distance of $\mathbf{x_c}$ to its epipolar line $\bar{\mathbf{l}}_\mathbf{c}$ which goes through the epipole $\mathbf{e_c}$ and the true point $\bar{\mathbf{x}}_\mathbf{c}$. The point $\mathbf{x}'_\mathbf{c}$ results from reflecting $\mathbf{x_c}$ through $\bar{\mathbf{x}}_\mathbf{c}$. The "reflected" distance d'_c and d_c differ only in the sign, i.e. $d'_c = -d_c$. Furthermore, $\mathbf{x}'_\mathbf{c}$ and $\mathbf{x_c}$ have identical propabilities, regardless whether the covariance ellipse is (a) circular or (b) elliptic.

the reflected point $\mathbf{x}'_\mathbf{c}$ has the same propability as the original point $\mathbf{x_c}$. Thus, the propability of d_l being positive is the same as being negative, in other words d_l is zero-mean. By the way, the function d_l is not point-symmetric: $d_l(\mathbf{n}) \neq -d_l(-\mathbf{n})$.

Where the covariance ellipse is actually elliptic, as in figure 5.13b, the zero-mean property of d_l is lost. Only in the special case where one of the ellipse's half axes is parallel to $\bar{\mathbf{l}}_\mathbf{l}$, d_l is still zero-mean. This is because the propability of d_l is not symmetric anymore.

We have just shown that the epipolar distance d_c in the current frame is zero-mean (regardless of the shape of the covariance ellipse), and that the epipolar distance d_l in the last frame is zero-mean if $Cov[\mathbf{n}] = \mathrm{diag}(\sigma^2_{(\mathbf{n})_1}, \sigma^2_{(\mathbf{n})_2}, 0)$. From experiments with simulated traffic scenes, we know that the correspondences produced by the optical flow algorithm approximately obey a zero-mean normal distribution with $Cov[\mathbf{n}] = \mathrm{diag}(0.55^2, 0.55^2, 0)$, i.e. the condition is fulfilled. The rooted symmetric epipolar distance J_{RSED} thus is zero-mean. The residual $r = C_p(J_{RSED})$ is also zero-mean, since C_p is the point-symmetric rooted Huber cost function.

All assumptions made in conjunction with equation 5.31 are fulfilled. Thus, equation 5.31 is used to compute the covariance matrix of the estimated ego-motion:

$$Cov[\hat{\mathbf{p}}_\mathbf{e}] = \frac{2\chi_{\min}}{N_e - 3}\mathbf{H}^{-T} \tag{5.34}$$

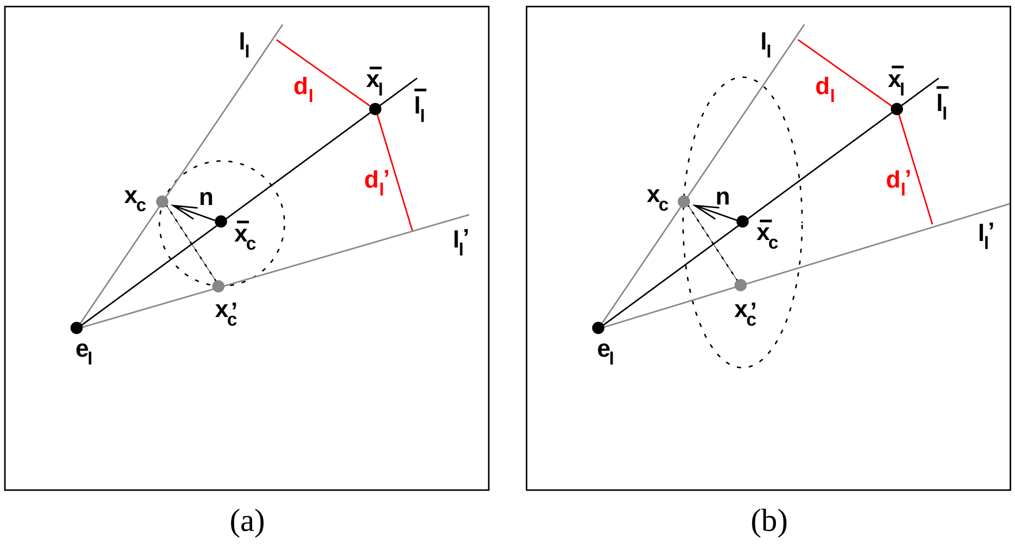


Figure 5.13: The epipolar distance d_l in the last frame is symmetric regarding the true epipolar line $\bar{\mathbf{l}}_\mathbf{l}$. This line is defined by the epipole $\mathbf{e_l}$ and the true point $\bar{\mathbf{x}}_\mathbf{c}$. For better visualization $\bar{\mathbf{l}}_\mathbf{l}$ goes through $\bar{\mathbf{x}}_\mathbf{c}$ (rotation-free ego-motion). The noise $\mathbf{n}$ in the measured point $\mathbf{x_c} = \bar{\mathbf{x}}_\mathbf{c} + \mathbf{n}$ changes $\bar{\mathbf{l}}_\mathbf{l}$ to $\mathbf{l_l}$ producing the epipolar distance d_l. The point $\mathbf{x}'_\mathbf{c}$ resulting from the reflection over $\bar{\mathbf{l}}_\mathbf{l}$ produces the epipolar distance d'_l. Both distances d'_l and d_l are equal except for the sign. In (a) the covariance ellipse is circular inducing identical propabilities for $\mathbf{x_c}$ and $\mathbf{x}'_\mathbf{c}$. This does not hold if the ellipse is elliptic (b).

5.6.3 Uncertainty of the Road Homography Estimate

In section 5.4, we recommended to employ the geometric error metric together with the known height of the camera when the road homography needs to be estimated. This approach led to the following estimate:

$$\hat{\mathbf{p}}_\mathbf{n} = \arg\min_{\mathbf{p_n}} \sum_{i=1}^{N_n} C_p((\mu_i)_1)^2 + C_p((\mu_i)_2)^2 \tag{5.35}$$

with $\mathbf{p_n} = (\alpha, \varphi)^T$ the parameter vector capturing the pitch angle α and the roll angle φ. Here the residuals are $r_i = C_p((\mu_i)_1) + C_p((\mu_i)_2)$ with μ the parallax vector.

In order to apply equation 5.31, we have to show that the assumptions made in conjunction with this equation are fulfilled. The terms $r_i \frac{\partial^2 r_i}{\partial\alpha\partial\varphi}$ are negligible w.r.t. the terms $\frac{\partial r_i}{\partial\alpha}\frac{\partial r_i}{\partial\varphi}$ (1st assumption) since the r_i's are zero-mean in the minimum (see 4th assumption for the reason). The correspondences (=$\mathbf{x}_i$'s) produced by the optical flow algorithm are independent (2nd assumption). The parallax vector μ is equal to the noise $\mathbf{n}$: $\mu = ((\mathbf{n})_1, (\mathbf{n})_2)^T$. Due to $\mathbf{n} \sim N(0, Cov[\mathbf{n}])$, and C_p point-symmetric the r_i's are identically distributed (3rd assumption) and zero-mean (4th assumption).

The assumptions are fulfilled thus equation 5.31 is applicable:

$$Cov[\hat{\mathbf{p}}_\mathbf{n}] = \frac{2\chi_{\min}}{2N_n - 2}\mathbf{H}^{-T} \tag{5.36}$$

5.6.4 Kalman Filtering

The road homography estimation gives us the normal vector of the road w.r.t. the camera. However, this vector may be inaccurate due to a low-textured road.

We take the ego-motion into account to improve the estimated road normal. Assuming a road of constant vertical slope, as shown in figure 5.10, the ego-motion expressed with the rotation matrix $\mathbf{R} = \mathbf{R}(\Delta\alpha, \Delta\psi, \Delta\varphi)$ represents the temporal derivative of the road normal, i.e. the normal vector at time instant k is the previous one at $k-1$ rotated by $\mathbf{R}_{k-1}$:

$$\mathbf{n}_k = \mathbf{R}_{k-1} \cdot \mathbf{n}_{k-1} \tag{5.37}$$

We now have two measurements of the road normal, first the estimate $\hat{\mathbf{n}}$ (built from $\hat{\mathbf{p}}_\mathbf{n}$), and second, the update rule (equation 5.37). Before we will combine them within a Kalman filter, we pay attention to the update rule. Due to the uncertainty in the estimated rotation $\hat{\mathbf{R}}$ (built from $\hat{\mathbf{p}}_\mathbf{e}$), the normal will drift away, if only this rule is applied. The update rule in conjunction with the estimated road normal $\hat{\mathbf{n}}$ prevents a drift. In situations where $\hat{\mathbf{n}}$ is poorly estimated, we need an alternative measurement: the average normal vector $\tilde{\mathbf{n}}$. It is learned online, employing a recursive low-pass: $\tilde{\mathbf{n}}_k = \lambda\tilde{\mathbf{n}}_{k-1} + (1-\lambda)\mathbf{n}_{k-1}$ with $\lambda \in (0,1)$.

The thoughts above lead to the following Kalman filter design, combining the estimates $\hat{\mathbf{R}}, \hat{\mathbf{n}}$ together with their corresponding covariance matrices $Cov[\hat{\mathbf{p}}_\mathbf{e}]$ and $Cov[\hat{\mathbf{p}}_\mathbf{n}]$ computed with the equations 5.34 and 5.36 respectively. The notation regarding the Kalman filter is taken from [Welch & Bishop 01].

- The process model reads: $\mathbf{x}_k = \mathbf{A}_{k-1} \cdot \mathbf{x}_{k-1} + \mathbf{w}_{k-1}$ where the state vector $\mathbf{x}$ represents the filtered normal vector. The state transition matrix is equal to the rotation matrix provided by the ego-motion estimation: $\mathbf{A} = \hat{\mathbf{R}}$.

- The process noise $\mathbf{w} \sim N(0, \mathbf{Q})$ reflects the uncertainty of the rotation and is characterized by the process covariance matrix $\mathbf{Q} = \mathbf{J}_\mathbf{e} Cov[\hat{\mathbf{p}}_\mathbf{e}]\, \mathbf{J}_\mathbf{e}^T$ with $\mathbf{J}_\mathbf{e} = \frac{\partial \mathbf{x}}{\partial \mathbf{p}_\mathbf{e}}$ the Jacobian matrix.

- The measurement model is

$$\mathbf{z}_k = \begin{pmatrix} \tilde{\mathbf{n}}_k \\ \hat{\mathbf{n}}_k \end{pmatrix} = \begin{pmatrix} \mathbf{x}_k \\ \mathbf{x}_k \end{pmatrix} + \mathbf{v}_k \tag{5.38}$$

  We have two "measurements" for the state $\mathbf{x}_k$. There is the average normal vector $\tilde{\mathbf{n}}_k$ and the estimated road normal $\hat{\mathbf{n}}_k$. The uncertainties of $\tilde{\mathbf{n}}_k$ and $\hat{\mathbf{n}}_k$ decide which measurement can be more trusted.

- The measurement noise $\mathbf{v} \sim N(0, \mathbf{R})$ characterized with the measurement covariance matrix[2] $\mathbf{R}$ consists of the variance of $\tilde{\mathbf{n}}_k$:

$$\sigma^2(\tilde{\mathbf{n}})_k = \lambda\sigma^2(\tilde{\mathbf{n}})_{k-1} + (1-\lambda)(\mathbf{n}_k - \tilde{\mathbf{n}}_k)^2 \tag{5.39}$$

[2]Sorry that the letter R is assigned to two distinctive entities.

and of the covariance of $\hat{\mathbf{n}}_k$:

$$Cov[\hat{\mathbf{n}}_k] = \frac{\partial \mathbf{n}}{\partial \mathbf{p_n}} Cov[\hat{\mathbf{p}}_\mathbf{n}] \left(\frac{\partial \mathbf{n}}{\partial \mathbf{p_n}} \right)^T \tag{5.40}$$

The measurement covariance matrix is:

$$\mathbf{R} = \begin{bmatrix} \operatorname{diag}\left(\sigma^2(\tilde{\mathbf{n}})_k\right) & 0 \\ 0 & Cov[\hat{\mathbf{n}}_k] \end{bmatrix} \tag{5.41}$$

When the Kalman filter performs the update step the state vector $\mathbf{x}$ will change its length. However, $\mathbf{x}$ represents a normal vector which should have a length of one. For this reason $\mathbf{x}$ is normalized to $||\mathbf{x}|| = 1$ after each update. The complete approach is summarized in algorithm 5.1.

5.6.5 Experimental Results

In this section, the Kalman filtering is tested on real traffic scenes. Three experiments are carried out.

Experiment 1

In order to visually compare the estimated road plane with the actual one, a straight road with a constant vertical slope is required. The vanishing point of the (parallel) road boundaries gives us one point on the horizon. The horizon of the estimated road plane should pass through that point. Figure 5.14 shows such a road. The image correspondences used for ego-motion and road normal estimation are shown in figure 5.14a. There are many correspondences on the road - thanks to a well textured road - allowing a good estimation of the road plane (yellow horizon line in 5.14b).

667 frames later the vehicle drives under a bridge causing a reduced illumination and therefore a low-textured road. Only a few correspondences are found on the road (figure 5.15a). This leads to a poor estimate of the road plane which can be seen in figure 5.15b. This situation corresponds to a high variance of the estimate depicted in figure 5.16. Around the frame 667 the variance is higher than normal indicating a poor estimate. In such a situation the Kalman filter updates the road normal incorporating mainly the ego-motion estimate. As a consequence the filtered road plane (red horizon line in fig.5.15b) compares well to the actual one. Beside the estimated and the filtered horizon, the integrated horizon is shown in figure 5.15b. It is the result of the integration of the estimated ego-motion from frame 0 to frame 667 using equation 5.37 together with an appropriate initial road normal. One can clearly see that the integral has drifted away.

The Kalman filtering effectively prevents a drift in the road normal and is able to cope with temporary lacks of texture on the road.

Algorithm 5.1 Road Homography Estimation and Filtering

Task: Estimate and filter the road homography given the ego-motion estimate $\hat{\mathbf{p}}_\mathbf{e}$ and correspondences $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$

1. Compute the driving corridor. Compute recursively the marginal points $\mathbf{x_{cor1}}^{(n)}$ and $\mathbf{x_{cor2}}^{(n)}$ for $n = 0..30$ (see section 5.1 for details). Apply the projection matrix $\mathbf{P_c}$ to get the images of these points. The closed polygon joining the points defines the driving corridor. Ω is the set of all image points inside the driving corridor.

2. Parameterize the road homography. The road homography $\mathbf{H_r} = \mathbf{K}(\mathbf{R} - \mathbf{e_c}\frac{\mathbf{n}^T}{h})\mathbf{K}^{-1}$ compatible with the ego-motion is parameterized by the normal vector $\mathbf{n} = \mathbf{R}(\alpha, 0, \varphi)(0, -1, 0)^T$ depending on the parameter vector $\mathbf{p_n} = (\alpha, \varphi)^T$.

3. Estimate the road homography. Use the parallax vector:

$$\mu = \begin{pmatrix} (\mathbf{x_c})_1 - \frac{(\mathbf{H_r x_l})_1}{(\mathbf{H_r x_l})_3} \\ (\mathbf{x_c})_2 - \frac{(\mathbf{H_r x_l})_2}{(\mathbf{H_r x_l})_3} \end{pmatrix}$$

to find the best estimate $\hat{\mathbf{p}}_\mathbf{n}$:

$$\hat{\mathbf{p}}_\mathbf{n} = \arg\min_{\mathbf{p_n}} \sum_{i=1}^{N_n} C_p((\mu(\mathbf{p_n}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_1)^2 + C_p((\mu(\mathbf{p_n}, \mathbf{x}_{\mathbf{l},i}, \mathbf{x}_{\mathbf{c},i}))_2)^2 \quad \text{with} \quad \mathbf{x_l} \in \Omega$$

with C_p the point-symmetric rooted Huber cost function. The functional is minimized by LM.

4. Compute the covariance matrices of the ego-motion estimate $Cov[\hat{\mathbf{p}}_\mathbf{e}]$ and the road normal estimate $Cov[\hat{\mathbf{p}}_\mathbf{n}]$ according to

$$\frac{2\chi_{\min}}{N-3}\mathbf{H}^{-T}$$

See section 5.6.1 for details.

5. Compute the average road normal and its variance.

average road normal: $\tilde{\mathbf{n}}_k = \lambda\tilde{\mathbf{n}}_{k-1} + (1-\lambda)\mathbf{n}_{k-1}$
variance: $\sigma^2(\tilde{\mathbf{n}})_k = \lambda\sigma^2(\tilde{\mathbf{n}})_{k-1} + (1-\lambda)(\mathbf{n}_k - \tilde{\mathbf{n}}_k)^2$

with $\lambda \in (0,1)$ and k the current time step.

6. Filter the road homography.
Feed the Kalman filter with the estimates: $\hat{\mathbf{p}}_\mathbf{e}$, $\hat{\mathbf{p}}_\mathbf{n}$, $\tilde{\mathbf{n}}$ and their uncertainties: $Cov[\hat{\mathbf{p}}_\mathbf{e}]$, $Cov[\hat{\mathbf{p}}_\mathbf{n}]$, $\sigma^2(\tilde{\mathbf{n}})$.
Update the Kalman filter. Then normalize the state vector $\mathbf{x}$ to $||\mathbf{x}|| = 1$. It represents the filtered road normal.

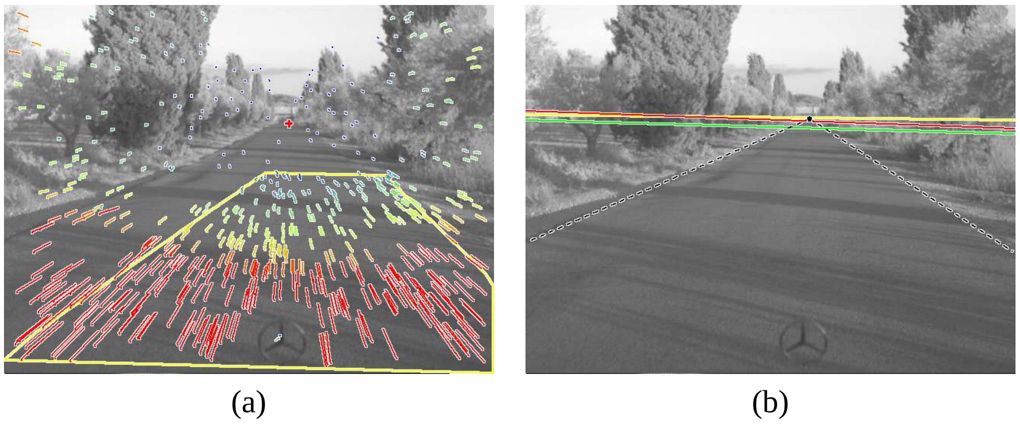

(a) (b)

Figure 5.14: Frame 20 of an image sequence containing a straight road with a constant vertical slope. The road is well textured. (a) The image correspondences outside the driving corridor (yellow area) are used to estimate the ego-motion. The correspondences inside the driving corridor are used to estimate the road normal. (b) The road normal represented by its yellow horizon line lies close to the vanishing point (black dot), i.e. it is well estimated. Also the integrated (green) and the filtered (red) horizons lie close to it.

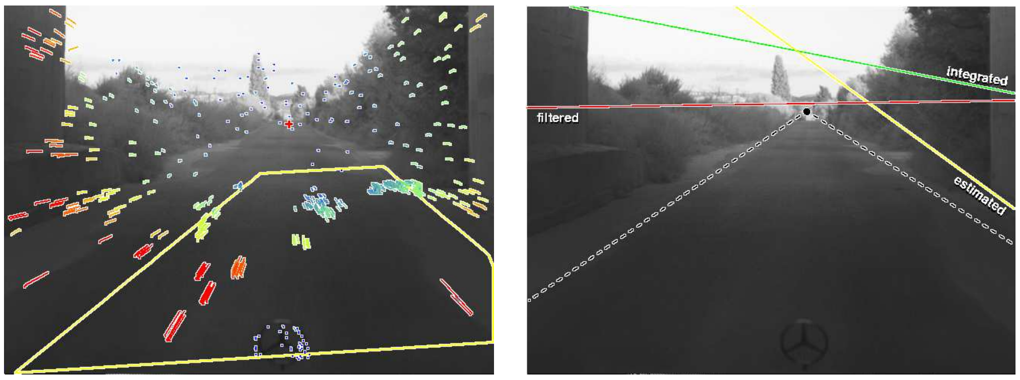
Experiment 2

In the second experiment, we generate a series of poor estimates of the road plane in order to investigate the filtering power. To this end, we take the last 100 frames of the straight road sequence and vary the number of correspondences N_n used for the road normal estimation. In each frame N_n correspondences are selected randomly from the set of measured correspondences. The lower N_n, the worse the estimate will be. We compare the estimated and filtered road normal to the ground truth. The cloud in the middle of the image serves as ground truth (fig. 5.17a). This object is immune to camera translations, since it is far away. Yaw and roll rotations shift the cloud horizontally and pitch rotations shift it vertically. Since the cloud's structure is mainly horizontal, horizontal shifts cannot be tracked very well. Thus, we concentrate only on vertical (pitch) motions. The vertical shift of the cloud is tracked using [Hager & Belhumeur 98].

Figure 5.17b shows the resulting standard deviations for error of the estimated and filtered pitch angle. The stabilising effect of the Kalman filter is evident. The error of the estimated pitch angle increases rapidly for $N_n < 10$ correspondences, whereas the error of the filtered pitch angle increases moderately.

Experiment 3

The proposed approach also works well if the ego-vehicle drives a curve. In figure 5.18 the vehicle just turned left at an intersection. In this situation, the lane markings are not straight which makes it unfeasible to extract the vanishing point. Here another special point is used as ground-truth information: the camera is mounted near the rearview mirror in the ego-vehicle. The rearview mirror of the car seen by the camera (fig. 5.18) has the same height above the road as the "ego-mirror". Any world point having the same height as the camera lies on the roads



(a) (b)

Figure 5.15: Frame 667 of the straight road sequence. The vehicle drives under a bridge causing a reduced illumination and therefore a low-textured road. (a) There are only some correspondences inside the driving corridor resulting in a poor estimate of the road normal (yellow horizon line in figure (b)), whereas the filtered road normal (red line) is still near the vanishing point (black dot). The integrated road normal (green line) has drifted away.

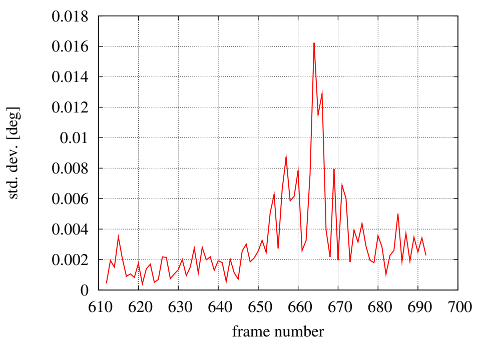
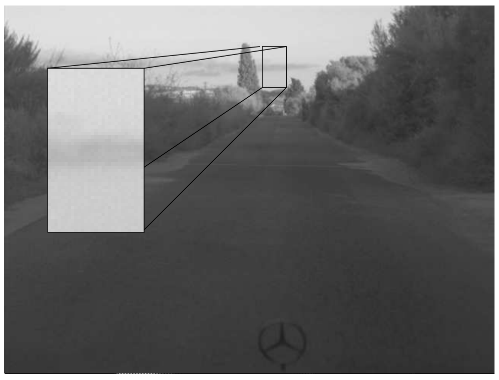


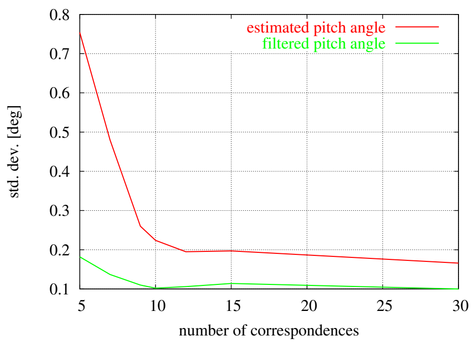
Figure 5.16: Standard deviation $\sqrt{(Cov[\hat{\mathbf{p}}_\mathbf{n}])_{11}}$ of the estimated pitch angle α computed with equation 5.36. Around frame 667 the road is low-textured causing higher values.

horizon line regardless of its depth. Furthermore, it does not matter whether the point is moving or not. This means that the horizon line should pass through the rearview mirror of the car. The filtered as well as the estimated horizon line lie very close to it. To see that the filtered horizon is correct whereas the estimated is not, we have to look at the C pillars. The filtered horizon line intersects both (c-säulen) at a same height, which does not hold for the estimated horizon line.

The entire approach - consisting of the ego-motion estimation, the road normal estimation, and the Kalman filtering - is very fast, since the computational expensive iterative minimizations are limited to a 3+2 parameter space. When 300 correspondences are used, the algorithm runs in about 2ms (Pentium IV 2.4GHz), excluding the computation of the correspondences.

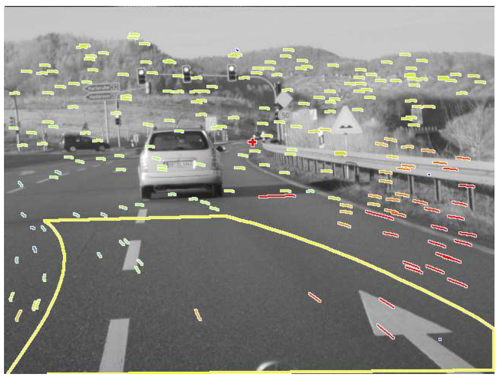


(a)

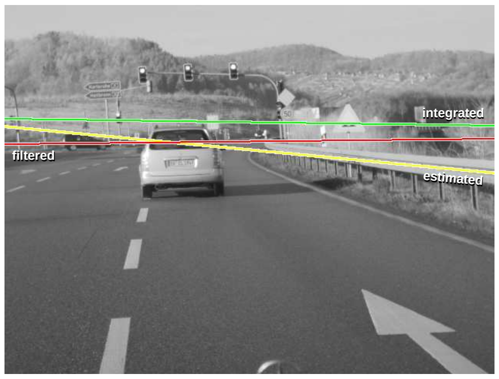


(b)

Figure 5.17: (a) The cloud in the background is tracked over 100 frames and serves as ground truth for the pitch angle. (b) Standard deviation for error of the estimated and filtered pitch angle depending on the number of correspondences used for the road normal estimation.



(a)



(b)

Figure 5.18: Frame 201 of the intersection sequence. The ego-vehicle turns left inducing a curved driving corridor (a). The correspondences inside the driving corridor are used for the road normal estimation. They are limited to 10 simulating a low-textured road. Figure (b) shows the resulting poor estimate of the road normal (yellow horizon line). The filtered road normal (red line) is in sane whereas the integrated road normal (green line) has drifted away.

Chapter 6

Detection of Independently Moving Objects

In the last chapters we had dealt with the estimation of the ego-motion and road homography. Why this effort? Well, we are now able to reconstruct the static part of the 3D scene, and we can put reconstructed 3D points into relation to the road. The reconstruction is the access point to the detection of moving objects: For 3D points which are actually static the reconstruction will be fine, but for 3D points which are moving the reconstruction will fail (in general). What does this mean?

A reconstructed 3D point has to fulfill certain constraints in order to be a valid static 3D point. If it violates any of them the 3D point is not static, hence it must move. These constraints play the essential role in the detection of moving objects.

In the following section the constraints for static 3D points are discussed. These constraints are well known to the computer vision community, but there is no algorithm which exploits them all. An algorithm doing so is introduced in section 6.3. It evaluates the constraints quantitatively in a unified manner. Experimental results in section 6.4 show its effectiveness. The points detected as moving must be grouped together to form broad objects. This clustering issue is pointed out in section 6.5. Although a lot of constraints for static 3D points exist there are some kinds of motion which (nearly) fulfill all constraints and thus are not detectable. These detection limits are investigated in section 6.6.

6.1 Constraints for Static 3D Points

In traffic scenes a static 3D point fulfills four constraints. The first three constraints apply for correspondences over two views. The fourth constraint is applicable if correspondences over three views are available. Each individual constraint raises the quality of detection.

- **Epipolar Constraint**
  The epipolar constraint expresses that the viewing rays of a static 3D point (the lines joining the projection centers and the 3D point) must meet. A moving 3D point in general induces skew viewing rays violating the constraint. Figure 6.1 illustrates it. This constraint we had already used when we estimated the ego-motion. The knowledge about the fundamental matrix is sufficient to evaluate this constraint.

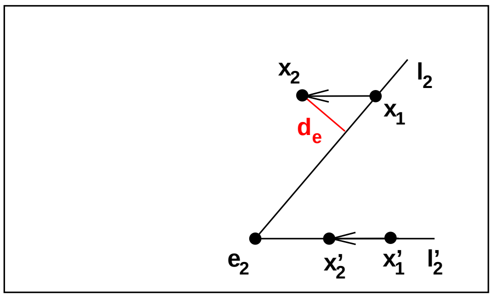


Figure 6.1: Epipolar constraint. The image of the second view is shown. The camera moves along its optical axis. An object moves lateral w.r.t. the camera inducing a horizontal optical flow shown by the correspondences $\mathbf{x_1} \leftrightarrow \mathbf{x_2}$ and $\mathbf{x}'_1 \leftrightarrow \mathbf{x}'_2$. The subscripts 1 and 2 denote entities in the first and the second view, respectively. $\mathbf{x_2}$ does not lie on the epipolar line $\mathbf{l_2}$ inducing the epipolar error d_e. $\mathbf{x}'_1$ moves along its epipolar line $\mathbf{l}'_2$ and thus fulfills the epipolar constraint. $\mathbf{e_2}$ is the epipole.

- **Positive Depth Constraint**
  The fact that all points seen by the camera must lie in front of it, is known as the positive depth constraint. It is also called cheirality constraint. If viewing rays intersect behind the camera, as in figure 6.2a, the actual 3D point must be moving. This constraint is independent of the scene structure. In order to evaluate it, the translation direction (forward or backward) of the camera has to be known, in addition to the Essential matrix.

- **Positive Height Constraint**
  All 3D points must lie above the road. If viewing rays intersect underneath the road, as in figure 6.2b, the actual 3D point must be moving. This constraint is not as powerful as the positive depth constraint since it applies only for image points under the horizon. Furthermore the geometry of the road has to be known. Commonly the road is approximated as a plane which is accurate enough in most cases. The driven distance between consecutive frames is also required, which is either retrieved with an odometer, or is extracted from the images directly using the measured optical flow of the road.

- **Trifocal Constraint**
  A triangulated 3D point utilizing the first two views, must triangulate to the same 3D point when the third view comes into consideration. This constraint is also called trilinear constraint. In figure 6.3 it is violated.

In traffic scenes no more constraints for static 3D points exist. In other applications there may be further constraints. In the field of robot indoor navigation, for example, the valid height is restricted due to the ceil. With the known height of the rooms a "maximum height constraint" is applicable.

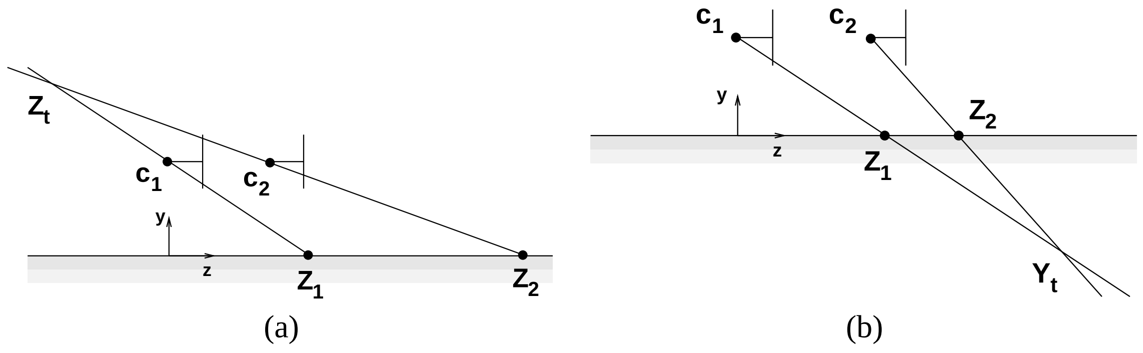


Figure 6.2: Side view: Positive depth (a) and positive height (b) constraint. The camera is moving from $\mathbf{c_1}$ to $\mathbf{c_2}$. A 3D point on the road is moving from Z_1 to Z_2. In (a) the travelled distance of the point is greater than the distance of the camera (overtaking object). The triangulated 3D point $\mathbf{Z_t}$ lies behind the camera, violating the positive depth constraint. In (b) the travelled distance of the point is smaller (preceding object). The triangulated 3D point $\mathbf{Y_t}$ lies underneath the road, violating the positive height constraint.

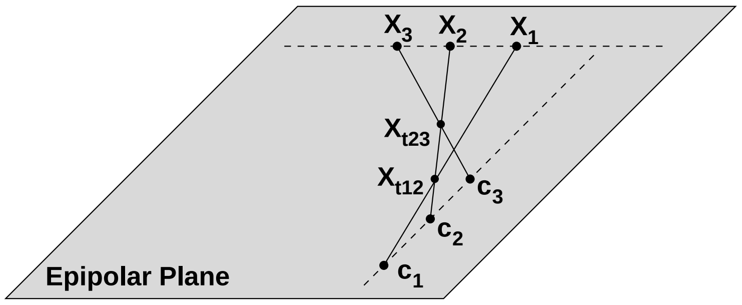


Figure 6.3: Trifocal Constraint. The camera observes a lateral moving 3D point ($\mathbf{X_1}$ to $\mathbf{X_3}$) while moving itself from $\mathbf{c_1}$ to $\mathbf{c_3}$. The triangulated point of the first two views is $\mathbf{X_{t12}}$. The triangulation of the last two views yields $\mathbf{X_{t23}}$ which does not coincide with $\mathbf{X_{t12}}$ violating the trifocal constraint.

6.2 Motion Detection Schemes in the Literature

The existing motion detection schemes exploit a subset of the constraints we have discussed in section 6.1 either directly or indirectly. In the following paragraphs three error metrics are described measuring the deviation from the constraints for static 3D points:

Cone criterion In [Wagner *et al.* 99] an error function for the ego-motion estimation utilizing the epipolar and the positive depth constraint is presented. Based on the "half-perspective" view, a conic error model is developed. An error cone is associated to a viewing ray. The apex of this cone coincides with the projection center of the camera, while the central vector is the viewing ray. The aperture angle ψ of the cone reflects the error ε of two corresponding viewing rays. ψ is the minimal angle where the intersection of corresponding error cones is not empty. If the the viewing rays intersect each other in front of both cameras, ψ is zero. However, if they intersect behind one camera, ψ is greater zero. In comparison, the epipolar geometry as a "full-perspective" approach would yield an error equal to zero.

Angle criterion The angle criterion uses the direction of the optical flow vectors. When moving purely translational towards the scene, all flow vectors are parallel to the corresponding epipolar lines and point away from the epipole (focus of expansion). This holds true for the entire static scene. We call this the *expected flow* direction. If this expectation is violated due to an independently moving object, the measured flow will deviate from the expected flow. Any camera rotations are removed in advance by applying the infinite homography.

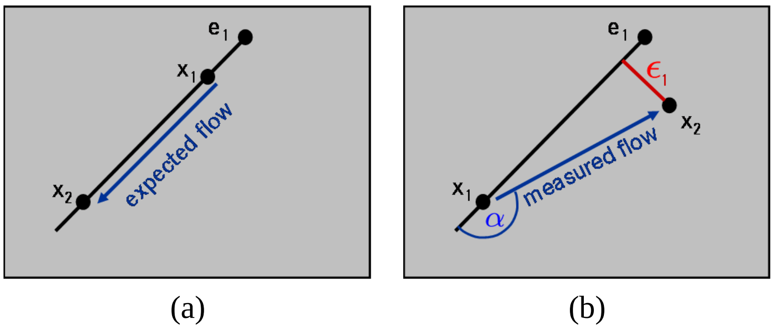


Figure 6.4: Angle criterion. (a) The image point of a static 3D point moves from $\mathbf{x_1}$ to $\mathbf{x_2}$ due to the camera motion towards the scene. The (expected) flow is parallel to the epipolar line and points away from the epipole $\mathbf{e_1}$. (b) The image point of a moving 3D point moves from $\mathbf{x_1}$ to $\mathbf{x_2}$. The measured flow has an angle α relative to the expected flow. In comparison, the epipolar error ε_1 is also depicted.

Figure 6.4(a) shows an example of the expected flow while figure 6.4(b) exemplifies a measured flow and the relation between the epipolar error and the angle error. A flow vector is classified as moving if the angle is greater than a certain threshold. This criterion requires a flow vector of sufficient length, since the angle is unstable for small flow vectors. In the event of zero flow an angle does not exist.

The angle criterion indirectly exploit the epipolar and the positive depth constraint. The incorporation of this criterion into a statistical framework is cumbersome due to its unfavourable properties: The angle does not fully correlate with the probability that a correspondence is actually moving. For example, if the correspondence obeys the epipolar constraint but not the positive depth constraint, the angle is always 180° regardless of the flow length. The work of [Woelk & Koch 04] employing the angle criterion in a Bayesian framework pays attention to that issue.

The angle criterion is also employed in [Pauwels & Hulle 04], and [Clauss *et al.* 05].

Planar motion parallax The parallax vector μ, defined as the deviation of the measured optical flow from the expected flow on the road plane (see equation 5.19), can be used to detect moving points. For correspondences violating the positive height constraint, the parallax vector points towards the epipole since the measured flow is shorter than expected. In [Giachetti *et al.* 98] and [Baehring *et al.* 05] the planar motion parallax is evaluated.

In section 6.5 we will discuss cluster algorithms exploiting the epipolar, the trifocal, or the multifocal constraint. They assign the correspondences to the distinct motions they find. The detection of the moving objects follows directly once the ego-motion is identified among the found motions. Commonly, the dominant motion, i.e. the motion with the highest number of correspondences, is supposed to be the ego-motion.

6.3 Error Metric Combining the Constraints

With the constraints in mind, the objective now is to measure quantitatively to which extent these constraints are violated. The resulting measurement function, called error metric, shall be correlated to the likelihood that the point is moving, i.e. higher values indicate a higher probability.

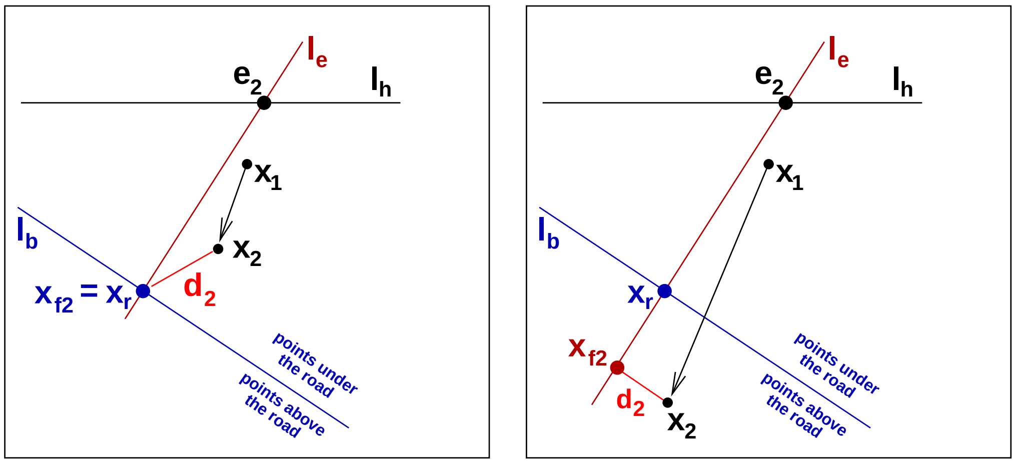
The error metric is developed in two steps. First, the two-view constraints are evaluated taking view one and two into account. Afterwards, the trifocal constraint is evaluated including the third view.

6.3.1 Two-view Constraints

The algorithm which is being developed combines the two-view constraints (epipolar, positive height, and positive depth constraint). An early version of it was published in [Klappstein *et al.* 06b]. The result of the algorithm is an error metric measuring the distance of the end point of a measured optical flow vector, to the nearest point which fulfills all constraints. The confidence of being a moving point is proportional to the error. In detail, the error increases with the skewness of the viewing rays and with the negative height of the triangulated 3D point. The error is also high for viewing rays meeting directly behind the camera.

The error is measured in units of pixel (no angles or other entities involved) allowing an easy incorporation into statistical evaluations. The geometric relations of the involved entities is depicted in figure 6.5. In the next section, when we work with three views, the notions "last frame" and "current frame" usually used in this thesis are not appropriate any more. For this reason we change the notions: the last frame becomes frame number one $(\mathbf{x_1}) = \mathbf{x_l}$, and the current frame becomes frame number two $(\mathbf{x_2} = \mathbf{x_c})$.

The measured flow vector starts in $\mathbf{x_1}$ (last frame) and ends in $\mathbf{x_2}$ (current frame). The start point $\mathbf{x_1}$ defines the epipolar line $\mathbf{l_e}$ going through the epipole $\mathbf{e_2}$. In the example shown in figure 6.5 $\mathbf{x_1}$ lies under the horizon line $\mathbf{l_h}$. Thus, the positive height constraint is applied: Assuming a forward moving camera, the point in the second frame matching perfectly with $\mathbf{x_1}$ and lying on the road is $\mathbf{x_r}$. The point $\mathbf{x_r}$ lies on the epipolar line and has zero height. Points on the epipolar line farther than $\mathbf{x_r}$ are above the road. They fulfill all constraints. Points closer to the epipole than $\mathbf{x_r}$ are under the road (violate the positive height constraint). The line $\mathbf{l_b}$ perpendicular to $\mathbf{l_e}$ defines the border line. In figure 6.5a the positive height constraint is violated. In this case the nearest point $\mathbf{x_{f2}}$ fulfilling all constraints is equal to the point on the road: $\mathbf{x_{f2}} = \mathbf{x_r}$. In figure 6.5b



(a) (b)

Figure 6.5: Two-view error. The second view is shown. The correspondence $\mathbf{x_1} \leftrightarrow \mathbf{x_2}$ violates the epipolar constraint. Additionally, in (a) the positive height constraint is violated. The point $\mathbf{x_{f2}}$ is the nearest point fulfilling all constraints. The two-view error d_2 measures the distance of $\mathbf{x_2}$ to that point. For detailed explanation see the text.

the positive height constraint is fulfilled. Here, $\mathbf{x_{f2}}$ lies at the foot of the perpendicular from the point $\mathbf{x_2}$. The *two-view error* d_2 is the distance from $\mathbf{x_2}$ to $\mathbf{x_{f2}}$.

For points $\mathbf{x_1}$ above the horizon line, the positive depth constraint applies. In that case the point on the road $\mathbf{x_r}$ is substituted by the point at infinity $\mathbf{x}_\infty$. This point perfectly matches with $\mathbf{x_1}$, when $\mathbf{x_1}$ is the image of an infinite 3D point. $\mathbf{x}_\infty$ also lies on the epipolar line. Points on the epipolar line farther than $\mathbf{x}_\infty$ are in front of the camera. The others lie behind it. The border line and the point $\mathbf{x_{f2}}$ are constructed analogue to the positive height constraint.

After this geometrical consideration we compute the two-view error. At first we need the horizon line $\mathbf{l_h}$. Its computation requires the rotation of the camera w.r.t. the road. This rotation we had estimated in chapter 5. The matrix $\mathbf{R_r} = \mathbf{R}(\alpha, 0, \varphi)$ rotates points from the road coordinate frame into the camera frame, where α and φ are the pitch angle and the roll angle of the road, respectively. With this information the vanishing points of the road's x-axis and z-axis can be computed:

$$\mathbf{v_x} = \mathbf{KR_r} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{v_z} = \mathbf{KR_r} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{6.1}$$

The line joining these two vanishing points results in the horizon line:

$$\mathbf{l_h} = \mathbf{v_x} \times \mathbf{v_z} \tag{6.2}$$

$$\mathbf{l_h} := \frac{\mathbf{l_h}}{(\mathbf{l_h})_3} \tag{6.3}$$

Equation 6.3 homogenizes the horizon line. This makes the next computation easier. The position of $\mathbf{x_1}$ decides which spatial constraint is applied. If $\mathbf{x_1}$ lies under the horizon, the positive height constraint is applied requiring the point on the road $\mathbf{x_r}$, otherwise the positive depth constraint is applied requiring the infinite point $\mathbf{x}_\infty$:

$$\mathbf{x_b} = \begin{cases} \mathbf{x_r} & , \mathbf{x_1}^T \mathbf{l_h} < 0 \\ \mathbf{x}_\infty & , \mathbf{x_1}^T \mathbf{l_h} \geq 0 \end{cases} \tag{6.4}$$

where $\mathbf{x_b}$ describes the generalized border point. The scalar product $\mathbf{x_1}^T \mathbf{l_h}$ is positive when $\mathbf{x_1}$ lies on the same side as the origin of the image (upper left corner: $(0,0,1)^T$). This is easily verified since $((0,0,1) \cdot \mathbf{l_h} = 1)$. It is assumed that the origin itself lies above the horizon.

The infinite point in equation 6.4 is computed via the infinite homography $\mathbf{H}_\infty = \mathbf{KRK}^{-1}$ mapping a point in the second frame onto the plane at infinity and back onto the image plane of the first frame:

$$\mathbf{x}_\infty = \mathbf{H}_\infty^{-1} \mathbf{x_1} \tag{6.5}$$

The road point also present in equation 6.4 is computed via the road homography $\mathbf{H_r}$. The estimation of $\mathbf{H_r}$ was the topic of chapter 5.

$$\mathbf{x_r} = \mathbf{H_r}^{-1} \mathbf{x_1} \tag{6.6}$$

Next we compute the border line $\mathbf{l_b}$. To this end we need the knowledge about the ego-motion we had obtained in chapter 4. The ego-motion reflects in the fundamental matrix: $\mathbf{F} = \mathbf{K}^{-T} [-\mathbf{Rt}]_\times \mathbf{RK}^{-1}$ where $\mathbf{K}$ was the calibration matrix, $\mathbf{t}$ the translation vector of the camera from the first frame to the second and $\mathbf{R}$ the rotation matrix of the second camera w.r.t to the first. The border line is perpendicular to the epipolar line $\mathbf{l_e}$ and goes through the border point $\mathbf{x_b}$:

$$\mathbf{l_b} = \begin{bmatrix} 0 & (\mathbf{x_b})_3 & 0 \\ -(\mathbf{x_b})_3 & 0 & 0 \\ (\mathbf{x_b})_2 & -(\mathbf{x_b})_1 & 0 \end{bmatrix} \mathbf{F}\mathbf{x_1} \tag{6.7}$$

The point $\mathbf{x_{f2}}$ fulfilling all constraints depends on the location of $\mathbf{x_2}$. If $\mathbf{x_2}$ lies on the same side of $\mathbf{l_b}$ as the epipole $\mathbf{e_2}$, then $\mathbf{x_{f2}}$ is equal to the border point, otherwise it lies at the foot of the perpendicular from $\mathbf{x_2}$:

$$\mathbf{x_{f2}} = \begin{cases} \mathbf{x_b} & , \mathbf{x_2}^T \mathbf{l_b} \cdot \mathbf{e_2}^T \mathbf{l_b} > 0 \\ \mathbf{d} \times \mathbf{x_2} \times \mathbf{l_e} & , else \end{cases} \tag{6.8}$$

with $\mathbf{d} = ((\mathbf{l_e})_1, (\mathbf{l_e})_2, 0)^T$. Remember that $\mathbf{l_b}$ is a homogeneous entity. Hence a single scalar product $\mathbf{x_2}^T \mathbf{l_b}$ is insufficient to check on which side $\mathbf{x_2}$ lies. Only together with the check for the epipole: $\mathbf{e_2}^T \mathbf{l_b}$ yields the desired result.

The final two-view error metric[1] is the distance from $\mathbf{x_2}$ to $\mathbf{x_{f2}}$:

$$d_2 = d(\mathbf{x_2}, \mathbf{x_{f2}}) \tag{6.9}$$

[1]To be honest, d_2 is a pseudometric since we may have $d(\mathbf{x_2}, \mathbf{x_{f2}}) = 0$ for distinct points $\mathbf{x_2} \neq \mathbf{x_{f2}}$.

6.3.2 Three-view Constraint

We now add the third view and consider the correspondence $\mathbf{x_1} \leftrightarrow \mathbf{x_2} \leftrightarrow \mathbf{x_3}$. As the point $\mathbf{x_{f2}}$ is defined such that it fulfills the two-view constraints, the reconstructed 3D point arising from the triangulation of the points $\mathbf{x_1}$ and $\mathbf{x_{f2}}$ constitute a valid 3D point. This 3D point is projected into the third view yielding $\mathbf{x_{f3}}$. The measured image point $\mathbf{x_3}$ will coincide with $\mathbf{x_{f3}}$ if the observed 3D point is actually static. Otherwise there is a distance d_3 (figure 6.6) between them which we call the *trifocal error*. $\mathbf{x_{f3}}$ is computed via the point-point-point transfer using the trifocal tensor [Hartley & Zisserman 03]. This fast approach avoids the explicit triangulation of the 3D point.

The overall error, combining the two-view constraints and the three-view constraint, is $d = d_2 + d_3$. It measures the minimal required displacement in pixels necessary to change a given correspondence into a correspondence belonging to a valid static 3D point. The higher d is the higher the likelihood is that the observed point is moving. The computation of d is summarized in algorihm 6.1.

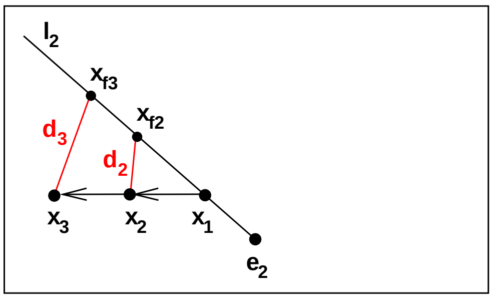


Figure 6.6: Trifocal error. The image of the second view is shown. The camera moves along its optical axis observing a lateral moving point $\mathbf{x_1} \leftrightarrow \mathbf{x_2} \leftrightarrow \mathbf{x_3}$. The closest point to $\mathbf{x_2}$ fulfilling the two-view constraints is $\mathbf{x_{f2}}$. The error arising from two-views is the distance d_2. Transfering the points $\mathbf{x_1}$ and $\mathbf{x_{f2}}$ into the third view yields $\mathbf{x_{f3}}$. If the observed 3D point was actually static its image $\mathbf{x_3}$ would coincide with $\mathbf{x_{f3}}$. However, the 3D point is moving which causes the trifocal error d_3. The overall error is $d = d_2 + d_3$. Note, that in general $\mathbf{x_1}$ and $\mathbf{x_{f3}}$ do not lie on the epipolar line $\mathbf{l_2}$.

The error metric relies on the optical flow which itself is uncertain in its measurement. To take this into account the error can be weighted by some entity representing the certainty of the measured optical flow. The weight function depends on the used optical flow algorithm. A simple weight function for example is the *corner response function* defined by Harris and Stevens [Harris & Stevens 88] measuring the "cornerness" of an image patch. A corner-like grey value structure is localized more accurately than a homogeneous structure resulting in a higher certainty of the optical flow. The flow algorithm used in this thesis (chapter 3) filters out non-corner-like structures. The resulting optical flow vectors have nearly the same accuracy. A weighting of the error would not have a significant benefit. There are other flow algorithms dealing not only with corner-like structures, but also with edge-like structures. An example is

Algorithm 6.1 Motion Detection

Task: Computation of the combined error of a correspondence given:

- a correspondence $\mathbf{x_1} \leftrightarrow \mathbf{x_2} \leftrightarrow \mathbf{x_3}$
- fundamental matrix $\mathbf{F}$ of the first and second view, defined by the ego-motion
- trifocal tensor $\mathcal{T}$, defined by the ego-motion
- road homography $\mathbf{H_r}$

1. Compute the horizon line. The vanishing points of the roads x-axis and z-axis are:

$$\mathbf{v_x} = \mathbf{KR_r}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \qquad \mathbf{v_z} = \mathbf{KR_r}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The horizon line then is:

$$\mathbf{l_h} = \mathbf{v_x} \times \mathbf{v_z}$$
$$\mathbf{l_h} := \frac{\mathbf{l_h}}{(\mathbf{l_h})_3}$$

2. Choose the border point. If $\mathbf{x_1}$ lies under the horizon the point on the road $\mathbf{x_r} = \mathbf{H_r}^{-1}\mathbf{x_1}$ is taken otherwise the infinite point $\mathbf{x_\infty} = \mathbf{H}_\infty^{-1}\mathbf{x_1}$:

$$\mathbf{x_b} = \begin{cases} \mathbf{x_r} & ,\mathbf{x_1}^T\mathbf{l_h} < 0 \\ \mathbf{x_\infty} & ,\mathbf{x_1}^T\mathbf{l_h} \geq 0 \end{cases}$$

3. Compute the border line.

$$\mathbf{l_b} = \begin{bmatrix} 0 & (\mathbf{x_b})_3 & 0 \\ -(\mathbf{x_b})_3 & 0 & 0 \\ (\mathbf{x_b})_2 & -(\mathbf{x_b})_1 & 0 \end{bmatrix} \mathbf{Fx_1}$$

4. Compute the point fulfilling the two-view constraints.

$$\mathbf{x_{f2}} = \begin{cases} \mathbf{x_b} & ,\mathbf{x_2}^T\mathbf{l_b} \cdot \mathbf{e_2}^T\mathbf{l_b} > 0 \\ \mathbf{d} \times \mathbf{x_2} \times \mathbf{l_e} & ,\mathit{else} \end{cases}$$

with $\mathbf{d} = ((\mathbf{l_e})_1, (\mathbf{l_e})_2, 0)^T$.

5. Compute the point fulfilling the three-view constraint. This is done using the trifocal tensor based point-point-point transfer. For details see [Hartley & Zisserman 03].

$$\mathbf{x_{f3}} = \mathbf{x_{f3}}(\mathbf{x_1}, \mathbf{x_{f2}}, \mathcal{T})$$

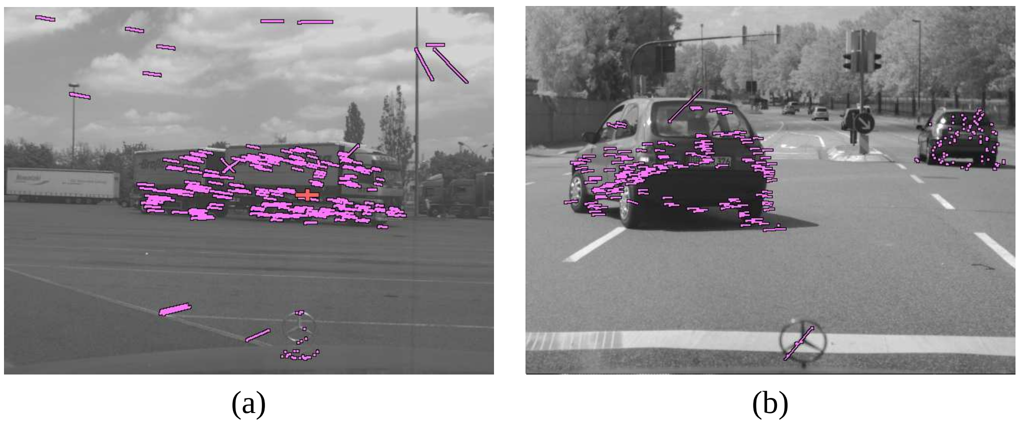
6. Compute the combined error.

$$d = d(\mathbf{x_2}, \mathbf{x_{f2}}) + d(\mathbf{x_3}, \mathbf{x_{f3}})$$

KLT [Tomasi & Kanade 91, Shi & Tomasi 94]. When using such an algorithm the weighting is highly beneficial.

6.4 Experimental Results

The motion detection algorithm developed last section is now applied to real imagery. To this end, all three algorithms 3.1, 5.1, and 6.1 are applied. The flow vectors are classified as the static environment or as a moving object according to their combined error d. A value of $T = 1.7$px is used as the threshold, according to the precision of the measured optical flow plus an additive safety margin. Figure 6.7 shows two traffic situations. Thanks to the exploitation of all constraints, almost all parts of the objects are detected as moving.



(a) (b)

Figure 6.7: Detection of moving points. (a) Crossing truck. (b) Preceding vehicles. The optical flow vectors shown are classified as moving. Few mismatched vectors occur in the sky and on the road. Due to visual clearness the number of vectors is reduced to one eighth.

6.5 Clustering

At this time we are able to detect moving 3D points based on the optical flow. But single moving points are insufficient to implement a robust driver assistence system. If one relies on single points, one misclassified point may cause a faulty reaction. For this reason the 3D points must be clustered to obtain broad objects. The task is to find the 3D points which belong to one and the same physical (moving) object (a vehicle, a pedestrian, ...). This is not easy to accomplish since an algorithm does not know how physical objects look. The only information an algorithm has are the 3D points.

In general a cluster algorithm searches for "common fates" among the input data. In our case 3D points, which are close together and which have a similar combined error, share the same fate. The likelihood that such points belong to one physical object is very high.

At this point a simple algorithm shall be discussed, although the clustering issue is beyond the scope of this thesis. The algorithm is based on the *connected component analysis* (CCA). The CCA clusters a binary image which is generated as follows.

We consider correspondences $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$ over two views and their two-view error d_2. If the correspondence was classified as moving, i.e. $d_2 > T = 1.7$, a one in the binary image is set at $\mathbf{x_c}$. Zeros are set where no correspondences were measured or where the correspondences were classified as static. An example is shown in figure 6.8. Once the binary image is made up,

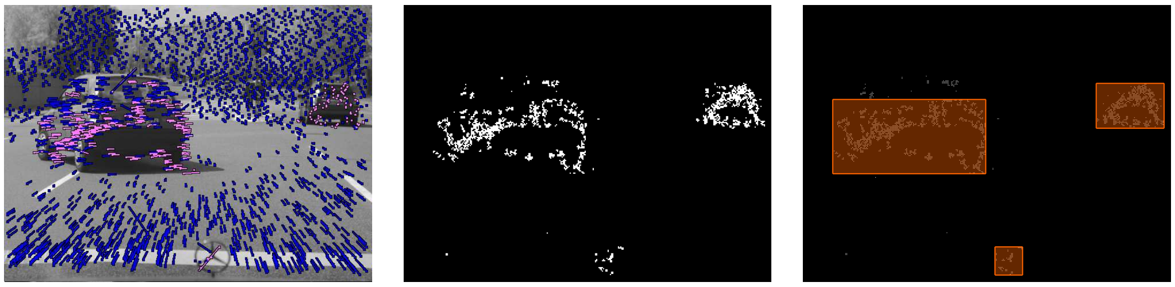

(a) (b) (c)

Figure 6.8: Clustering using CCA. (a) Blue flow vectors are classified as static, the magenta ones as moving. Due to visual clearness the number of vectors is reduced to one eighth. (b) Binary image. (c) Bounding boxes of the clusters.

the CCA goes through it pixel by pixel. If a one is found it looks whether there is a cluster in the neighbourhood of the current position. If yes, the current position is attached to this cluster, otherwise a new cluster is spawned. This approach is very efficient since one pass is enough to cluster to the image. The outcome of the algorithm is shown in figure 6.9.

Note, that this algorithm does not take all the available information into account. Beside the check for spatial vicinity, one could additionally check for similarities in the optical flow. Correspondences having a similar displacement $\mathbf{x_c} - \mathbf{x_l}$ (inhomogeneous points here) probably belong to one object. Since $\mathbf{x_l}$ remains unused in the CCA algorithm its performance is limited.

The literature has two-view cluster algorithms utilizing $\mathbf{x_l}$ and $\mathbf{x_c}$. However, these algorithms only rely on a subset of the available constraints. There is no algorithm taking full advantage of all constraints. This remains as future research. The existing cluster algorithms not only estimate the ego-motion, but also the motions of the moving objects. In this multibody motion estimation concept it is not differentiated between the ego-motion and the motions of the objects. Indeed, for clustering purposes it is not necessary to know which motion is the motion caused by the ego-vehicle. The task of finding the different motions given a set of correspondences is tackled mainly in three ways:

Multibody epipolar constraint The epipolar constraint for multiple objects is made up by multiplying the single epipolar constraints:

$$\prod_{i=1}^{N} \mathbf{x_c}^T \mathbf{F}_i \mathbf{x_l} = 0 \tag{6.10}$$

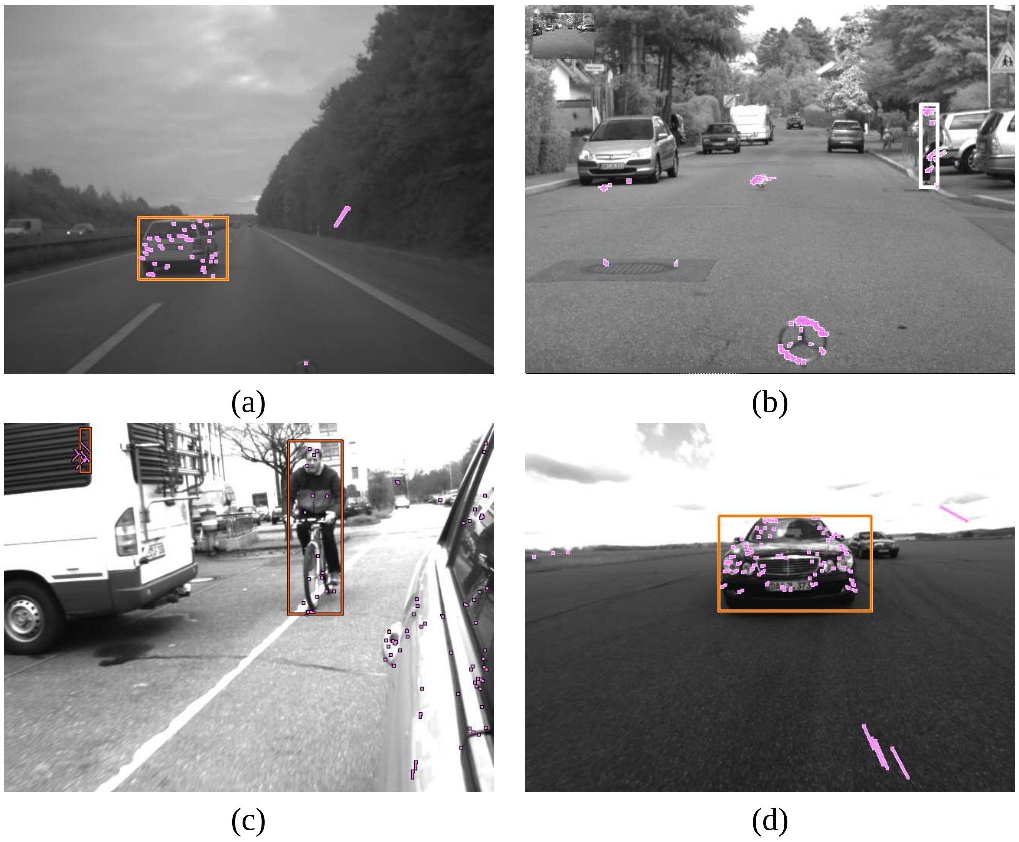


Figure 6.9: Results of motion detection + clustering. The shown flow vectors were classified as moving. The rectangles denote the bounding boxes of the clusters. (a) cut-in vehicle. (b) running child. (c) cyclist within the blind spot. The flow vectors on the ego-vehicle are moving, too. They were not incorporated into the clustering. (d) follower just before rear crash.

Each fundamental matrix $\mathbf{F}_i$ encodes one motion. Suppose we have two motions in the image, for example the ego-motion and the motion caused by an independently moving object (IMO). In particular, we have two fundamental matrices ($\mathbf{F}_1$ and $\mathbf{F}_2$) we are searching for, not knowing to which fundamental matrix the correspondences belong. The multibody epipolar constraint 6.10 is fulfilled regardless of the motion the correspondence $\mathbf{x_l} \leftrightarrow \mathbf{x_c}$ belongs to. If it belongs to the first motion, then $\underbrace{\mathbf{x_c}^T \mathbf{F}_1 \mathbf{x_l}}_{=0} \cdot \underbrace{\mathbf{x_c}^T \mathbf{F}_2 \mathbf{x_l}}_{\neq 0} = 0$. Otherwise, if it belongs to the second motion, the second factor would be zero.

Each correspondence gives rise to one instance of constraint 6.10. Having a sufficient number of correspondences the constraint 6.10 is decomposable into the distinct fundamental matrices [Ma *et al.* 04]. Once the $\mathbf{F}_i$'s are identified, the individual correspondences are assigned to the fundamental matrix which mostly fulfills the epipolar constraint. This approach was extended to three views, resulting in the multibody trifocal constraint [Hartley & Vidal 04].

Multibody factorization The multibody factorization is a multiple view approach aiming at the decomposition of a huge matrix $\mathbf{W}$ containing the correspondences. $\mathbf{W}$ is decomposed into a product of two matrices $\mathbf{W} = \mathbf{MS}$ separating the motion parameters contained in the motion matrix $\mathbf{M}$ from the 3D points contained in the shape matrix $\mathbf{S}$.

Suppose we have an orthographic camera and a 3D point $\mathbf{x_w}$ moving relative to it. The projection of this 3D point at time instant k is given by:

$$\mathbf{x}_k = \begin{pmatrix} (\mathbf{x}_k)_1 \\ (\mathbf{x}_k)_2 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_k & \mathbf{t}_k \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x_w} \tag{6.11}$$

Note, that $\mathbf{x}_k$ is an inhomogeneous point. $\mathbf{R}_k$ and $\mathbf{t}_k$ denote the rotation and translation of the 3D point at time instant k. Suppose that we have N correspondences over F frames, and that we collect all the measurements into a single matrix:

$$\mathbf{W} = \begin{bmatrix} (\mathbf{x}_{1,1})_1 & \cdots & (\mathbf{x}_{1,N})_1 \\ \vdots & & \vdots \\ (\mathbf{x}_{F,1})_1 & \cdots & (\mathbf{x}_{F,N})_1 \\ (\mathbf{x}_{1,1})_2 & \cdots & (\mathbf{x}_{1,N})_2 \\ \vdots & & \vdots \\ (\mathbf{x}_{F,1})_2 & \cdots & (\mathbf{x}_{F,N})_2 \end{bmatrix} = \begin{bmatrix} \mathbf{i}_1^T & (\mathbf{t}_1)_1 \\ \vdots & \vdots \\ \mathbf{i}_F^T & (\mathbf{t}_F)_1 \\ \mathbf{j}_1^T & (\mathbf{t}_1)_2 \\ \vdots & \vdots \\ \mathbf{j}_F^T & (\mathbf{t}_F)_2 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{x}_{\mathbf{w}1} & \cdots & \mathbf{x}_{\mathbf{w}N} \end{bmatrix} = \mathbf{MS} \tag{6.12}$$

with $\mathbf{i}_k^T$ and $\mathbf{j}_k^T$ the first and second row of the k-th rotation matrix. The factorization is done using the singular value decomposition and exploiting the fact that the vectors $\mathbf{i}_k^T$ and $\mathbf{j}_k^T$ are orthogonal.

When two motions are present and when the correspondences are sorted, the shape matrix $\mathbf{S}$ takes on a block diagonal form:

$$\mathbf{W}^\star = [\mathbf{M}_1|\mathbf{M}_2] \cdot \begin{bmatrix} \mathbf{S}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_2 \end{bmatrix} \tag{6.13}$$

The task is to find a column permutation of $\mathbf{W}$ determining the canonical form $\mathbf{W}^\star$. Once $\mathbf{W}^\star$ is known the segmentation is done. All 3D points contained in $\mathbf{S_1}$ move according to $\mathbf{M_1}$ and belong to the first object. The other 3D points in $\mathbf{S_2}$ belong to the second object. For details on the column permutation of $\mathbf{W}$ refer to [Costeira & Kanade 98]. The multibody factorization method was recently extended to perspective cameras [Vidal 05].

Expectation Maximization Expectation Maximization (EM) alternates between motion estimation and clustering. Given an initial set of clusters the motion of each cluster is estimated. The results are used to refine the clusters. With refined clusters the motions are estimated again, and so on, until the solutions converge. [Torr 98] employs this approach and further selects the appropriate motion model out of four models: fundamental matrix, affine fundamental matrix, homography, and affinity.

6.6 Detection Limit

The experimental results we have seen last section are promising. Each moving object was detected. But when we look at figure 6.10 we get disappointed. The car in front of the ego-

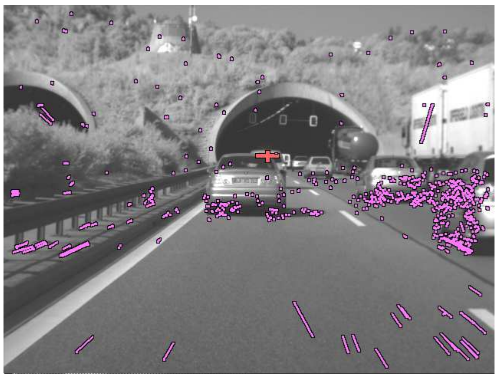


Figure 6.10: Detection of preceding objects. Only the very bottom part of the car in front of the ego-vehicle is detected. The cars on the right hand side are detected to a higher extent.

vehicle is hardly detected. This raises the question, utilizing the different constraints for static 3D points, which kinds of motion are detectable and to which extent?

The answers we give here were also published in [Klappstein *et al.* 07b]. In order to detect a moving object reliably, the error metric developed in section 6.3 must be greater than a certain threshold T, whereas the threshold should reflect the noise in the correspondences (optical flow). A reasonable choise is $T = 3\sigma$ with σ the standard deviation of the correspondences.

In the following we consider the three most frequent kinds of motion in traffic: parallel, lateral and circular motion. We model the motion of the camera and the object as shown in figure 6.11. It is not necessary to investigate camera rotations about its projection center, since they do not influence the detection limit. One can always compensate these rotations by a virtual inverse rotation.

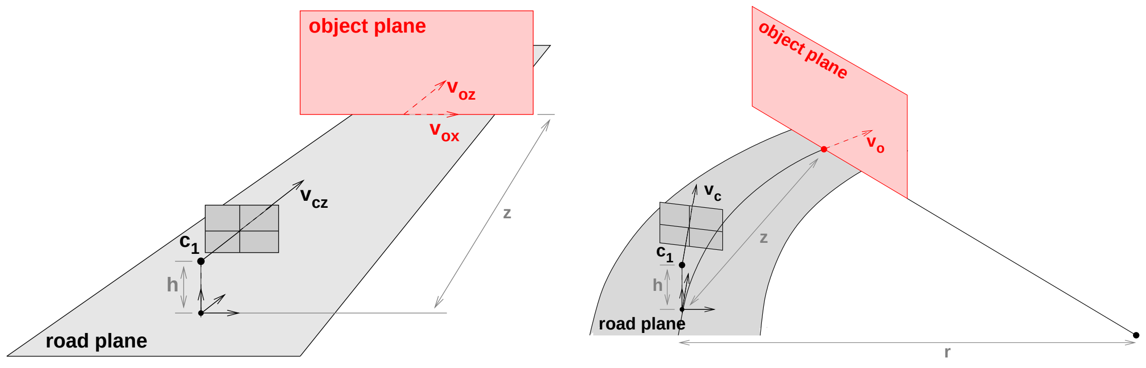
6.6.1 Linear Motion

The detection limits for the linear motions (parallel and lateral motion) are illustrated by means of three examples:

1. Overtaking object: The object moves parallel to the camera but faster.
   $v_{\text{cz}} = 30\text{km/h}$, $v_{\text{oz}} = 40\text{km/h}$, $v_{\text{ox}} = 0\text{km/h}$

2. Preceding object: The object moves parallel to the camera but slower.
   $v_{\text{cz}} = 30\text{km/h}$, $v_{\text{oz}} = 20\text{km/h}$, $v_{\text{ox}} = 0\text{km/h}$

3. Crossing object: The object moves lateral to the camera.
   $v_{\text{cz}} = 30\text{km/h}$, $v_{\text{oz}} = 0\text{km/h}$, $v_{\text{ox}} = -5\text{km/h}$



(a) (b)

Figure 6.11: Motion model utilized for the investigation of the detection limit. The cameras projection center in the first view is $\mathbf{c_1}$. The moving object is modelled as a plane. (a) Linear motion: The (object)plane moves parallel (w.r.t. the camera) with speed v_{oz} and lateral with speed v_{ox}. The distance of the camera to the object is z, to the road it is h. The camera moves along its optical axis with speed v_{cz}. (b) Circular motion: Both, camera and object, move along a circle with radius r. The tangential speed of the camera is v_{c}, that of the object is v_{o}.

The subscripts stand for: c = camera, o = object, z = longitudinal direction, x = lateral direction. Anti-parallel motion ($v_{\mathrm{cz}} > 0$km/h, $v_{\mathrm{oz}} < 0$km/h, $v_{\mathrm{ox}} = 0$km/h) is not detectable. This issue is addressed in subsection 6.6.4. In the examples, other important parameters are: focal length $f = 1000$px, principal point $(x_0, y_0) = (320, 240)$, height of camera above the road $h = 1$m, distance to object $z = 20$m, time between consecutive frames $\Delta t = 40$ms.

The detection limits of the linear motions are shown in figure 6.12. Each image shows the first view. Inside the black regions the error metric is lower than $T = 0.5$px (assuming a std. dev. in the correspondences of $\sigma = 0.167$px). Parts of the object seen in these regions are not detected as moving. There is one important point in the image: the *point of collision*. This is the point where the camera will collide with the object, provided that the object is slower than the camera. We will see that this dangerous point is not detectable in many cases.

The first row of figure 6.12 considers the epipolar constraint only. As can be seen, parallel motion is not detected at all. Lateral motion is detected to a high extent. The black region is shaped like a bow tie.

In the second row of figure 6.12, the positive depth constraint is added. Overtaking objects are now detected. The error metric in this case is identical to the motion parallax induced by the plane at infinity. The optical flow of points at infinity is zero (camera does not rotate). Thus, the motion parallax is equal to the length of the measured optical flow. The contour lines (lines where the error metric takes on a constant value) are circular around the epipole. Preceding objects are still not detected. In the case of lateral motion the bow tie is cracked. The motion is also detected between the epipole and the point of collision due to the violation of the positive depth constraint.

The use of the positive height constraint (third row of figure 6.12) gains the power of detection for the image part below the horizon. In the case of parallel motion (overtaking and preceding

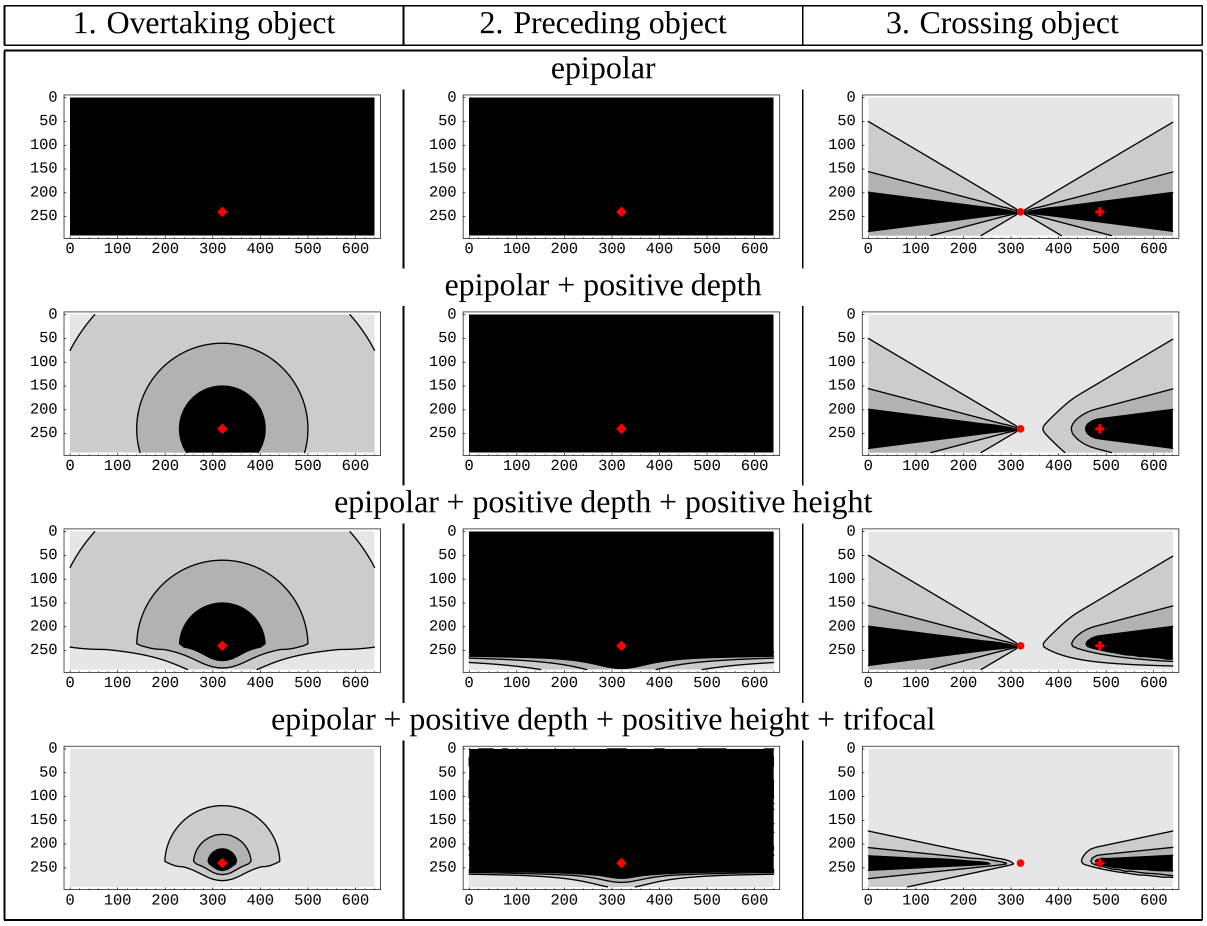


Figure 6.12: Detection limits for different kinds of linear motion and constraints. The images show the first view (compare to fig. 6.11). They are truncated at row 290, since below this row there is no object but the road. Inside the black regions the motion is not detected. The contour lines $2T$ and $4T$ are shown, too. The red point marks the epipole, the red cross is the point of collision. Further explanation is given in the text.

objects) the error metric below the horizon is identical to the motion parallax induced by the road plane. It is possible to detect preceding objects but it is a challenging task. Lateral motion only benefits from the positive height constraint only on the right-hand side of the epipole.

Adding the trifocal constraint yields the best achievable results. The parallel motion profits mainly from the larger driven distance of the camera, since the camera moves from $\mathbf{c}_1$ to $\mathbf{c}_3$ (not just to $\mathbf{c}_2$). This just increases the signal to noise ratio. Similar results would be obtained if only the first and the third view would be evaluated. This does not hold for the lateral motion. The trifocal constraint also allows detection to the left of the epipole.

The reason for that is given in figure 6.3 on page 101. There the camera moves from $\mathbf{c}_1$ to $\mathbf{c}_3$ observing a point moving from $\mathbf{X}_1$ to $\mathbf{X}_3$. A situation is chosen such that the trajectories of the camera and the point are co-planar. They move within the epipolar plane. Considering the first two views, the two-view constraints are fulfilled. The viewing rays meet perfectly at the point $\mathbf{X}_{t12}$. This point lies in front of the cameras and above the road. Consequently, this kind of motion is not detected over two views alone. Taking the third view into account reveals the motion, since the triangulated point $\mathbf{X}_{t23}$ of the second and third view is different from $\mathbf{X}_{t12}$.

We have seen that in case of the linear motion the strength of the trifocal constraint is not very high. The trifocal constraint shows its strength if the cameras translational direction changes over time, as is the case with circular motion.

6.6.2 Circular Motion

The circular motion is modelled as shown in figure 6.11b. To demonstrate the detection limit for this case we consider an example similar to the "preceding object" example: $v_c = 30$km/h, $v_o = 20$km/h, $z = 20$m, and $r = 100$m.

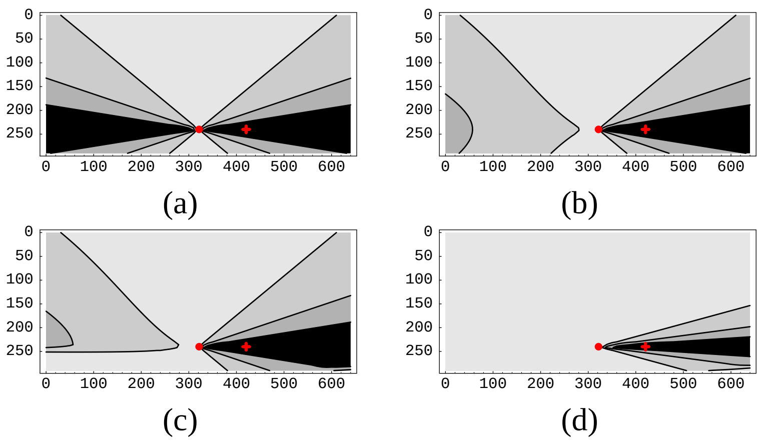


Figure 6.13: Detection limit in the case of circular motion. The images show the first view (compare to fig. 6.11b). They are truncated at row 274, since below there is no object but the road. Inside the black regions the motion is not detected. The contour lines $2T$ and $4T$ are also shown. The red point marks the epipole, the red cross is the point of collision. (a) Epipolar constraint. (b) + positive depth constraint. (c) + positive height constraint. (d) + trifocal constraint.

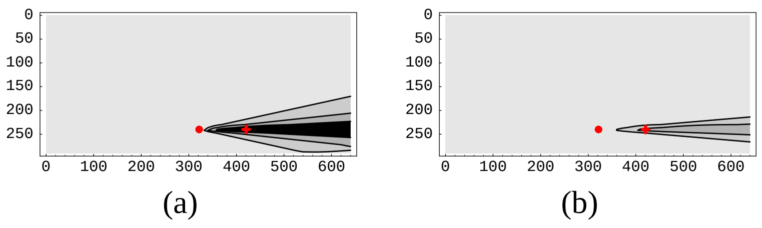


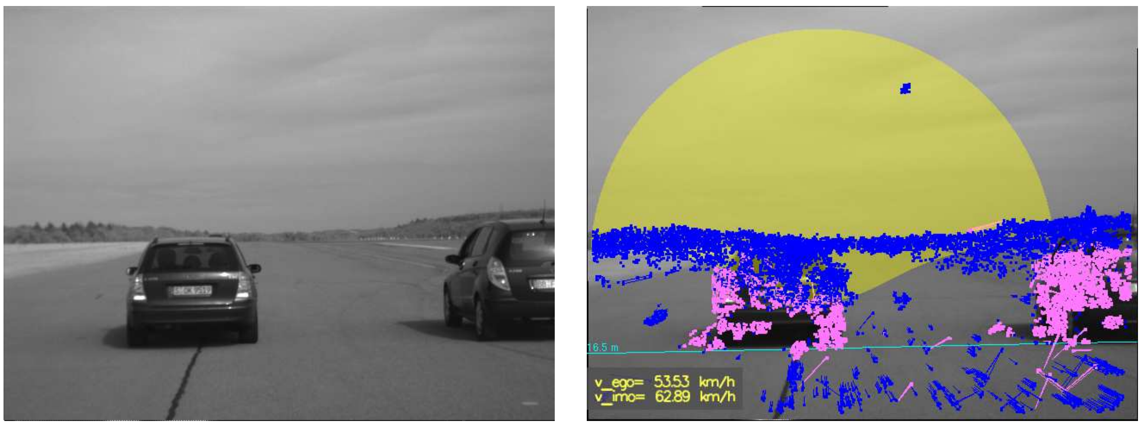
Figure 6.14: Detection limit in the case of circular motion with tripled time period Δt compared to figure 6.13. (a) Epipolar + positive depth + positive height constraint. (b) + trifocal constraint.

Figure 6.13 shows the detection limit. Although the object is slower than the camera, which was a problem for the parallel motion case, the circular motion is detected to a high extent (fig. 6.13a). With the positive depth constraint taken into account, the entire region to the left of the epipole is detected. It seems that the trifocal constraint (fig. 6.13d) just shrinks the black region, meaning that it only improves the signal to noise ratio. However, this is not true. If we triple the time period $\Delta t = 120$ms the black region vanishes (figure 6.14b). Consequently, the entire object is detected as moving and so is the point of collision. The power of the two-view constraints is insufficient to detect that point.

Taking more than three views into account just increases the signal to noise ratio and hence shrinks the black regions but does not change the shapes of the contour lines (unless camera and object accelerate differently).

6.6.3 Experimental Verification

In this section we apply the study on the detection limit to real imagery. Furthermore, we detect the moving objects based on the measured optical flow and the proposed error metric d_2. The detection result is compared to the theoretical detection limit.



(a) (b)

Figure 6.15: Experimental verification. (a) Original image with two moving vehicles in front. (b) The semi-transparent yellow region shows the image region where the motion is not detectable. The measured optical flow vectors are classified as static (blue / dark) and moving (magenta / bright).

Figure 6.15a shows two vehicles driving in front of the camera (ego-vehicle). They are faster than the camera and move parallel to it. First, the detection limit is computed. To this end, the distance to the objects and the speed of them are required. The on-board radar sensor provides this information: $z = 16.5$m and $v_{oz} = 62.9$km/h. The speed of the camera, retrieved by odometry, is $v_{cz} = 53.5$km/h. With this information, together with the camera calibration, the non-detectable region computes to that shown in figure 6.15b. Thereby the two-view constraints are considered.

The actual detection of the vehicles is carried out by the evaluation of the two-view error metric d_2 utilizing the measured optical flow. Radar data is ignored. Flow vectors with $d_2 >$

$T = 1.7$px are classified as moving. The result is shown in figure 6.15b. One can see that the theoretical detection limit matches well to the practical one.

The vehicle on the right side is completely detected whereas only the lower part of the vehicle in the middle of the image is detected.

6.6.4 Issue of Anti-Parallel Motion

Oncoming objects on a straight road constitute an anti-parallel motion. There is no way to detect this motion by means of the constraints for static 3D points. An inherent ambiguity prevents this. The moving object also could be a static object with smaller size and shorter depth. We call such a static pendant a phantom object (see figure 6.16(a)).

Only a heuristic approach enables the detection of such motion [Klappstein *et al.* 06a]. We can assume that any object in the world is opaque and stands on the ground. The latter one is violated for traffic signs, since in most situations it is difficult to measure any optical flow on the pole. Hence, the traffic sign seems to hover over the ground. Vehicles, however, are almost completely present in the optical flow, due to their cuboidal form.

With this heuristic we are able to distinguish between the seeming static object and the moving one: The assumptions create a zone behind the phantom object in which no other object is allowed to be present. If there is a triangulated point within that zone the phantom object is revealed (see figure 6.16(b)).

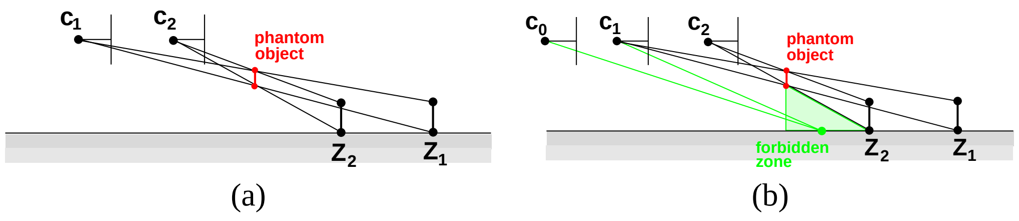


Figure 6.16: Anti-parallel motion. (a) The camera moves from $\mathbf{c_1}$ to $\mathbf{c_2}$ observing the pole moving from Z_1 to Z_2. The triangulation provides a hovering phantom object which is closer than the orignal one. (b) The green zone is prohibited. If there is a measured point within this zone, such as the green point, the phantom object is revealed as a moving object.

The algorithm evaluating the region under the object (forbidden zone) is now developed. Figure 6.17a shows an oncoming vehicle. After the ego-motion and the road homography were estimated and moving objects were detected we look for static objects. An efficient method detecting them is the evaluation of the planar motion parallax which we had already met when we estimated the road homography, see equation 5.19. Static objects do not belong to the road, hence their parallax is significantly higher than zero. Figure 6.17b shows this. In the next step, the flow vectors are clustered using the CCA from section 6.5. The cluster shown in figure 6.17b forms the phantom object.

We now need the depth of the phantom object in order to compute the forbidden zone. Here the direct linear transform (DLT) triangulation method [Hartley & Zisserman 03] is employed getting the depth of one correpondence. Although this method does not constitute a MLE it

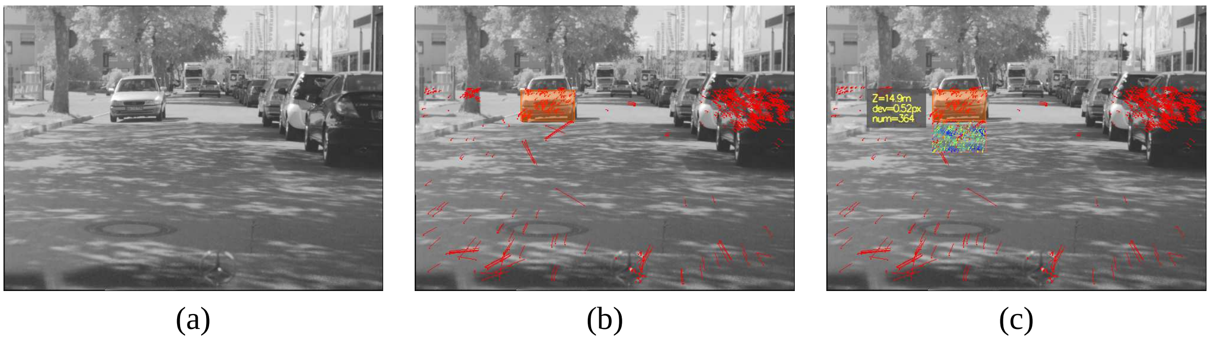


Figure 6.17: Detection of anti-parallel moving objects. (a) An oncoming vehicle. (b) Clustered vehicle. The parallax of the shown correspondences is higher than 2 pixels. (c) The forbidden zone under the vehicle contains a piece of the road. The correspondences are color-coded by their parallaxes. The color goes from blue to red representing a parallax of 0 and 2 pixels, respectively. The zone contains 364 correspondences with a median parallax of 0.52 pixels. The object is revealed as a moving object.

is accurate enough and easy to compute. The median depth of all correspondences inside the phantom object is a robust estimate of its depth.

Using the depth, the bottom line of the object is now projected onto the road, forming the forbidden zone. If this zone contains a piece of the road the object is revealed as a moving object. Whether there is road or not is found out by evaluating the parallax again. If the median parallax falls below a certain threshold (e.g. 1 pixel) the zone is considered as road. In figure 6.17c this is the case.

This approach requires a well-textured road. When no correspondences are available inside the forbidden zone the ambiguity cannot be resolved.

6.6.5 Summary

The investigation of detection limits for independently moving objects revealed that:

- Objects which are faster than the camera are detected to a higher extent than those which are slower. That is a pity because slower objects are the dangerous ones. We will not collide with a faster object.
- In the event of linear motion, the dangerous point of collision is not detected at all, what an irony of fate!
- The trifocal constraint exhibits its potential if the motion of the camera is circular (non-linear). Then the point of collision is detectable (in principle).
- Anti-parallel moving objects are not detected at all by means of the constraints for static 3D points. A heuristic approach helps to detect such motion.

Chapter 7

Summary and Outlook

Summary

In this thesis the detection of moving objects in traffic scenes based on the optical flow has been investigated. To this end, the flow vectors belonging to the static scene must be separated from the flow vectors on the moving objects. This separation relies on the four constraints a valid static 3D point obeys. In this thesis these constraints were named. Further constraints supplying the detection of single moving points do not exist. A novel algorithm was developed combining the constraints in a unified manner. The resulting error metric measures the minimal displacement required to change a given correspondence into a correspondence representing a valid static 3D point. The formulation of the error metric in the image domain allows an easy incorporation into a statistical framework, i.e. when the uncertainty in the measured optical flow is considered.

The detectability of moving objects was investigated and it was found out that in the event of linear motion the dangerous point of collision is not detected. In practice, this means the smaller the image of an object being on a collision course, the more difficult its detection. Crossing objects as occuring at intersections and objects driving parallel to and faster than the ego-vehicle (overtaking objects) are detected to a high extent. In contrast, objects driving parallel to and slower than the ego-vehicle (objects which are overtaken) are hardly detected.

In case of non-linear motion, e.g. circular, the point of collision is detectable provided that the time period of observation is sufficiently long. This means when cornering objects are detectable even if they are slower than the ego-vehicle. Oncoming objects on a straight road (anti-parallel motion) are only detectable if the heuristic, which was introduced in this thesis, is applied.

In order to compute the error metric measuring the deviation from the constraints for static 3D points, the knowledge about the ego-motion and the location of the camera relative to the road plane is required. A known approach estimating the ego-motion was extended by a motion model. It was shown that the estimation became considerably more robust. The known error metric J_{SED} as well as the Huber cost function were changed slightly so that the assumptions made by the Levenberg-Marquardt minimization are fulfilled. This helped saving time needed for the minimization. It was found out that not all image regions contribute similarly to the estimate. In particular, it was hinted that the yaw rate is estimated poorly if the camera is mounted at 90°

angle w.r.t. the vehicle’s longitudinal axis.

The location of the camera relative to the road plane consists of the normal vector of the road plane, and the height of the camera above the road. An algorithm was developed estimating the road normal using the planar motion parallax. In contrast to the ego-motion estimation, here one is faced with poorly localized correspondences in cases of a low-textured road. For this reason, a Kalman filter was designed which is able to cope with temporary drop outs of the estimation.

Outlook

Clustering

The algorithms developed in this thesis constitute a robust system for the detection of moving points. The clustering of the detected moving points to objects was addressed briefly. In particular, the CCA algorithm was discussed. There are more sophisticated cluster algorithms in the literature, for example *graph cut* [Boykov & Veksler 05] or the *level set method* [Sethian 99, Aubert & Kornprobst 02], which perform the clustering by minimization of an energy functional. All cluster algorithms are recipes describing the food preparation but not the ingredients. The latter ones, meaning the input data, are problem specific. In case of level set the question is how to formulate the energy functional. In case of graph cut the question is how to deploy the graph. These questions have a severe impact on the performance of the clustering and are not trivial to answer. A first work using graph cut to cluster the detected moving points exists [Gruber 08].

Optical Flow

In this thesis an optical flow algorithm was used computing the optical flow over two consecutive frames. An optical flow algorithm which is able to track local image features over several frames is advantageous, as the time of observation is increased. An increased time of observation involves an increased driven distance of the ego-vehicle. Higher driven distances benefits the detectability of moving objects. If an optical flow algorithm with tracking capability is applied the question raises how to evaluate the constraints for static 3D points in a recursive fashion.

The ego-motion estimation benefits from higher driven distances, too. How can be the ego-motion estimated recursively? Online SLAM methods do so (page 48). However, they involve the estimation of nuisance parameters, namely 3D points. Is there a way to avoid this in favour of a reduced computational burden?

Appendix A

Rotation Matrices in $\mathbb{R}^3$

A rotation matrix rotates the coordinate system. The matrix $\mathbf{R}(x,0,0)$ for example rotates the coordinate system about the x-axis through an angle x measured in rad. The rotation sequence:

$$\mathbf{R}(x,y,z) = \mathbf{R}(x,0,0) \cdot \mathbf{R}(0,y,0) \cdot \mathbf{R}(0,0,z)$$

first rotates the coordinate system about the z-axis. Then it is rotated about the rotated y-axis. Finally it is rotated about the twice rotated x-axis. Such a sequence is also called *Euler sequence*.

Alternatively, the same sequence can be treated as a rotation about fixed axes. In that case the coordinate system is first rotated about the x-axis, then rotated about the original y-axis, and finally about the original z-axis.

If one wants to rotate points instead of the coordinate system one has to apply the inverse rotation $\mathbf{R}^{-1}$. Since rotation matrices are ortho-normal the inverse is equal to the transposed matrix: $\mathbf{R}^{-1} = \mathbf{R}^T$.

The rows of the rotation matrix show how the unit vectors are rotated. For example, the third row indicates the new z-axis. Reason: The point $(0,0,1)^T$ representing the z-unit-vector is rotated according to $\mathbf{R}^T(0,0,1)^T = ((r)_{31},(r)_{32},(r)_{33})^T$.

- $\mathbf{R}(x,0,0) =$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos x & \sin x \\ 0 & -\sin x & \cos x \end{bmatrix}$$

- $\mathbf{R}(0,y,0) =$

$$\begin{bmatrix} \cos y & 0 & -\sin y \\ 0 & 1 & 0 \\ \sin y & 0 & \cos y \end{bmatrix}$$

- $\mathbf{R}(0,0,z) =$

$$\begin{bmatrix} \cos z & \sin z & 0 \\ -\sin z & \cos z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $\mathbf{R}(x,y,z) =$

$$\begin{bmatrix} \cos y \cos z & \cos y \sin z & -\sin y \\ \cos z \sin x \sin y - \cos x \sin z & \cos x \cos z + \sin x \sin y \sin z & \cos y \sin x \\ \cos x \cos z \sin y + \sin x \sin z & -\cos z \sin x + \cos x \sin y \sin z & \cos x \cos y \end{bmatrix}$$

- $\mathbf{R}(0,0,z) \cdot \mathbf{R}(0,y,0) \cdot \mathbf{R}(x,0,0) =$

$$\begin{bmatrix} \cos y \cos z & \cos z \sin x \sin y + \cos x \sin z & -\cos x \cos z \sin y + \sin x \sin z \\ -\cos y \sin z & \cos x \cos z - \sin x \sin y \sin z & \cos z \sin x + \cos x \sin y \sin z \\ \sin y & -\cos y \sin x & \cos x \cos y \end{bmatrix}$$

- $\mathbf{R}(0,y,0) \cdot \mathbf{R}(x,0,0) \cdot \mathbf{R}(0,0,z) =$

$$\begin{bmatrix} \cos y \cos z - \sin x \sin y \sin z & \cos z \sin x \sin y + \cos y \sin z & -\cos x \sin y \\ -\cos x \sin z & \cos x \cos z & \sin x \\ \cos z \sin y + \cos y \sin x \sin z & -\cos y \cos z \sin x + \sin y \sin z & \cos x \cos y \end{bmatrix}$$

In case of very small rotation angles the trigonometric functions can be approximated by the first order term of their Taylor series: $\cos x \approx 1$ and $\sin x \approx x$. Applying this and setting the bilinear and trilinear monomials zero ($xy = xz = yz = xyz = 0$) yields the linearized rotation matrix:

$$\mathbf{R}_{\text{lin}} = \mathbf{I} + \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix} \right]_{\times} = \begin{bmatrix} 1 & z & -y \\ -z & 1 & x \\ y & -x & 1 \end{bmatrix}$$

The rotation order does not matter here. The multiplication of linearized rotation matrices is commutative.

Appendix B

Miscellaneous

B.1 Calibration Matrix and its Inverse

The calibration matrix captures the intrinsic camera parameters:

- focal length in horizontal direction: f_x
- focal length in vertical direction: f_y
- horizontal component of the principal point: x_0
- vertical component of the principal point: y_0

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Often the inverse of the calibration matrix is needed. It is easily computed:

$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{x_0}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{y_0}{f_y} \\ 0 & 0 & 1 \end{bmatrix}$$

B.2 Projection Matrix and its Inverse

The projection matrix of a finite perspective camera is composed by the calibration matrix **K**, the rotation matrix **R**, and the translation **t**:

$$\mathbf{P} = \mathbf{KR}[\mathbf{I}| - \mathbf{t}] \tag{B.1}$$

P is a 3×4 matrix and thus not invertable. However, the (Moore-Penrose) pseudo-inverse $\mathbf{P}^+$ can be applied. $\mathbf{P}^+$ is defined such that $\mathbf{PP}^+ = \mathbf{I}$. For quadratic matrices the pseudo-inverse is equal the common inverse. In general $(\mathbf{AB})^+ \neq \mathbf{B}^+\mathbf{A}^+$. Nevertheless, it holds in the case of the projection matrix. Thus the pseudo-inverse is given by:

$$\mathbf{P}^+ = [\mathbf{I}|-\mathbf{t}]^+\mathbf{R}^T\mathbf{K}^{-1} \tag{B.2}$$

with

$$[\mathbf{I}|-\mathbf{t}]^+ = \frac{1}{1+\mathbf{t}^T\mathbf{t}} \begin{bmatrix} 1+(\mathbf{t})_2^2+(\mathbf{t})_3^2 & -(\mathbf{t})_1(\mathbf{t})_2 & -(\mathbf{t})_1(\mathbf{t})_3 \\ -(\mathbf{t})_1(\mathbf{t})_2 & 1+(\mathbf{t})_1^2+(\mathbf{t})_3^2 & -(\mathbf{t})_2(\mathbf{t})_3 \\ -(\mathbf{t})_1(\mathbf{t})_3 & -(\mathbf{t})_2(\mathbf{t})_3 & 1+(\mathbf{t})_1^2+(\mathbf{t})_2^2 \\ -(\mathbf{t})_1 & -(\mathbf{t})_2 & -(\mathbf{t})_3 \end{bmatrix} \tag{B.3}$$

B.3 Cross Product Matrix

The cross product of two three-dimensional vectors **a** and **b** may be expressed in terms of a 3×3 skew-symmetric matrix:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -(\mathbf{a})_3 & (\mathbf{a})_2 \\ (\mathbf{a})_3 & 0 & -(\mathbf{a})_1 \\ -(\mathbf{a})_2 & (\mathbf{a})_1 & 0 \end{bmatrix} \tag{B.4}$$

The cross product then reads:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \left(\mathbf{a}^T[\mathbf{b}]_\times\right)^T \tag{B.5}$$

Bibliography

[Armangué *et al.* 02] Xavier Armangué, Helder Araújo, and Joaquim Salvi, *Differential epipolar constraint in mobile robot egomotion estimation*, IEEE International Conference on Pattern Recognition (ICPR) (Québec, Canada), 2002, pp. 599–602.

[Aubert & Kornprobst 02] Gilles Aubert and Pierre Kornprobst, *Mathematical problems in image processing*, Springer, 2002.

[Avidan & Shashua 00] Shai Avidan and Amnon Shashua, *Trajectory triangulation: 3d reconstruction of moving points from a monocular image sequence*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), no. 4, 348–357.

[Baehring *et al.* 05] Dietrich Baehring, Stephan Simon, Wolfgang Niehsen, and Christoph Stiller, *Detection of close cut-in and overtaking vehicles for driver assistance based on planar parallax*, IEEE Intelligent Vehicles Symposium, 2005.

[Baumela *et al.* 00] L. Baumela, L. Agapito, P. Bustos, and I. Reid, *Motion estimation using the differential epipolar equation*, 15th International Conference on Pattern Recognition, vol. 3, 2000, pp. 840–843.

[Boykov & Veksler 05] Yuri Boykov and Olga Veksler, *Graph cuts in vision and graphics: Theories and applications*, Handbook of Mathematical Models in Computer Vision (Nikos Paragios, Yunmei Chen, and Olivier Faugeras, eds.), Springer, 2005, pp. 79 – 96.

[Bruss & Horn 81] A.R. Bruss and B.K.P. Horn, *Passive navigation*, MIT AI Memo, 1981.

[Chalimbaud *et al.* 05] Pierre Chalimbaud, François Berry, François Marmoiton, and Serge Alizon, *Design of a hybrid visuo-inertial smart sensor*, Workshop on Integration of Vision and Inertial Sensors in conjunction with IEEE International Conference on Robotics and Automation (ICRA), 2005.

[Clauss *et al.* 05] Martin Clauss, Pierre Bayerl, and Heiko Neumann, *Segmentation of independently moving objects using a maximum-likelihood principle*, Autonome Mobile Systeme (Stuttgart, Germany), Springer, 2005.

[Costeira & Kanade 98] J. P. Costeira and T. Kanade, *A multibody factorization method for independently moving objects*, International Journal of Computer Vision **29** (1998), no. 3, 159–179.

[Domke & Aloimonos 06] Justin Domke and Yiannis Aloimonos, *A probabilistic notion of camera geometry: Calibrated vs. uncalibrated*, Photogrammetric Computer Vision (PCV), 2006.

[Engels *et al.* 06] Chris Engels, Henrik Stewénius, and David Nistér, *Bundle adjustment rules*, Photogrammetric Computer Vision (PCV), 2006.

[Faugeras & Luong 01] Oliver Faugeras and Quang-Tuan Luong, *The geometry of multiple images*, The MIT Press, 2001.

[Fischler & Bolles 81] M. Fischler and R. Bolles, *Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography*, Commun. Assoc. Comp. **24** (1981), 381–395.

[Franke *et al.* 05] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan Gehrig, *6D-vision: Fusion of stereo and motion for robust environment perception*, Proceedings of the 27th DAGM Pattern Recognition Symposium (Vienna, Austria), Springer, 2005.

[Gay 83] D. M. Gay, *Algorithm 611 – subroutines for unconstrained minimization using a model trust-region approach*, ACM Trans. Math. Software **9** (1983), 503–524.

[Gehrig 05] Stefan K. Gehrig, *Large-field-of-view stereo for automotive applications*, Omnidirectional Vision (OmniVis), October 2005.

[Giachetti *et al.* 98] Andrea Giachetti, Marco Campani, and Vincent Torre, *The use of optical flow for road navigation*, IEEE Transactions on Robotics and Automation **14** (1998), no. 1.

[Gibson 50] James Jerome Gibson, *The perception of the visual world*, Houghton Mifflin, 1950.

[Gruber 08] Daniel Gruber, *Segmentierung querbewegter Hindernisse mittels Graph Cut*, Master's thesis, University of Konstanz, Germany, 2008.

[Hager & Belhumeur 98] G. Hager and P. Belhumeur, *Efficient region tracking with parametric models of geometry and illumination*, IEEE Transactions on Pattern Analysis and Machine Intelligence (1998), 1025–1039.

[Harris & Stevens 88] C.G. Harris and M.J. Stevens, *A combined edge and corner detector*, Proc. of 4th Alvey Vision Conference, 1988.

[Hartley & Vidal 04] Richard Hartley and René Vidal, *The multibody trifocal tensor: Motion segmentation from 3 perspective views*, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2004.

[Hartley & Zisserman 03] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, second ed., Cambridge Press, 2003.

[Haussecker & Spies 99] H.W. Haussecker and H. Spies, *Motion*, Handbook of Computer Vision and Applications (B. Jähne, H.W. Haussecker, and P. Geissler, eds.), Academic Press, 1999, pp. 309–396.

[Heeger & Jepson 92] David J. Heeger and Allan D. Jepson, *Linear subspace methods for recovering translation direction*, RBCV-TR, 1992.

[Hillenbrand 07] Jörg Hillenbrand, *Fahrerassistenz zur Kollisionsvermeidung*, Ph.D. thesis, University of Karlsruhe, Germany, 2007.

[Huber 81] Peter J. Huber, *Robust statistics*, Wiley, 1981.

[Irani *et al.* 97] Michal Irani, Benny Rousso, and Shmuel Peleg, *Recovery of ego-motion using region alignment*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997), no. 3, 268–272.

[Jähne 05] Bernd Jähne, *Digital image processing*, 6th ed., Springer, 2005.

[Jepson & Heeger 90] Allan D. Jepson and David J. Heeger, *Subspace methods for recovering rigid motion, part II: Theory*, Technical Reports on RBCV-TR, 1990.

[Ke & Kanade 03] Qifa Ke and Takeo Kanade, *Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection*, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2003, pp. I–390– I–397.

[Klappstein *et al.* 06a] Jens Klappstein, Fridtjof Stein, and Uwe Franke, *Flussbasierte Eigenbewegungsschätzung und Detektion von fremdbewegten Objekten*, Workshop Fahrerassistenzsysteme (FAS), 2006.

[Klappstein *et al.* 06b] Jens Klappstein, Fridtjof Stein, and Uwe Franke, *Monocular motion detection using spatial constraints in a unified manner*, IEEE Intelligent Vehicles Symposium (IV), 2006.

[Klappstein *et al.* 07a] Jens Klappstein, Fridtjof Stein, and Uwe Franke, *Applying kalman filtering to road homography estimation*, Workshop on Planning, Perception and Navigation for Intelligent Vehicles in conjunction with IEEE International Conference on Robotics and Automation (ICRA), 2007.

[Klappstein *et al.* 07b] Jens Klappstein, Fridtjof Stein, and Uwe Franke, *Detectability of moving objects using correspondences over two and three frames*, Pattern Recognition (Proc. DAGM) (Heidelberg, Germany), LNCS, Springer, 2007, pp. 112–121.

[Lang & Pinz 05] Peter Lang and Axel Pinz, *Calibration of hybrid vision / inertial tracking systems*, Workshop on Integration of Vision and Inertial Sensors in conjunction with IEEE International Conference on Robotics and Automation (ICRA), 2005.

[Langer & Mann 04] Michael S. Langer and Richard Mann, *On the computation of image motion and heading in a 3-d cluttered scene*, Optical Flow and Beyond, Kluwer Academic Press, 2004.

[Lawn & Cipolla 96] Jonathan Lawn and Roberto Cipolla, *Reliable extraction of the camera motion using constraints on the epipole*, ECCV96, 1996, pp. II:161–173.

[Litwiller 05] David Litwiller, *CMOS vs. CCD: Maturing technologies, maturing markets*, Photonics Spectra Magazine (2005).

[Lobo & Dias 05] Jorge Lobo and Jorge Dias, *Relative pose calibration between visual and inertial sensors*, Workshop on Integration of Vision and Inertial Sensors in conjunction with IEEE International Conference on Robotics and Automation (ICRA), 2005.

[LonguetHiggins & Prazdny 80] H.C. Longuet-Higgins and K. Prazdny, *The interpretation of a moving retinal image*, RoyalP **B-208** (1980), 385–397.

[LonguetHiggins 81] H.C. Longuet-Higgins, *A computer algorithm for reconstructing a scene from two projections*, Nature (1981), 133–135.

[Lourakis & Argyros 05] Manolis I.A. Lourakis and Antonis A. Argyros, *Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?*, International Conference on Computer Vision (ICCV), 2005.

[Lowe 99] David G. Lowe, *Object recognition from local scale-invariant features*, IEEE International Conference on Computer Vision (ICCV), 1999, pp. 1150–1157.

[Ma *et al.* 04] Yi Ma, Stefano Soatto, Jana Košecká, and S. Shankar Sastry, *An invitation to 3-d vision*, Springer Verlag, 2004.

[Makadia *et al.* 05] Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis, *Radon-based structure from motion without correspondences*, CVPR, 2005.

[Mandelbaum *et al.* 98] R. Mandelbaum, G. Salgian, and H. Sawhney, *Correlation-based estimation of ego-motion and structure from motion and stereo*, Proceedings of the International Conference on Computer Vision, 1998.

[Mann & Langer 05] Richard Mann and Michael S. Langer, *Spectrum analysis of motion parallax in a 3d cluttered scene and application to ego-motion*, Journal of the Optical Society of America A (2005), 1717–1731.

[Maronna *et al.* 06] Ricardo A. Maronna, Douglas R. Martin, and Victor J. Yohai, *Robust statistics: Theory and methods*, Wiley, 2006.

[Maybank 92] Stephen Maybank, *Theory of reconstruction from image motion*, Springer, 1992.

[Michaelsen *et al.* 06] Eckart Michaelsen, Wolfgang von Hansen, Michael Kirchhof, Jochen Meidow, and Uwe Stilla, *Estimating the essential matrix: Goodsac versus ransac*, Photogrammetric Computer Vision (PCV), 2006.

[Molton *et al.* 04] N. Molton, A. Davison, and I. Reid, *Locally planar patch features for real-time structure from motion*, British Machine Vision Conference (BMVC), 2004.

[Mosteller & Tukey 77] Frederick Mosteller and John W. Tukey, *Data analysis and regression: A second course in statistics*, Addison-Wesley, 1977.

[Nistér 03] David Nistér, *Preemptive ransac for live structure and motion estimation*, Proceedings of the International Conference on Computer Vision, 2003.

[Nistér 04] David Nistér, *An efficient solution to the five-point relative pose problem*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (2004), 756–770.

[Pauwels & Hulle 04] Karl Pauwels and Marc M. Van Hulle, *Segmenting independently moving objects from egomotion flow fields*, Early Cognitive Vision Workshop, 2004.

[Poelman & Kanade 97] Conrad Poelman and Takeo Kanade, *A paraperspective factorization method for shape and motion recovery*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) **19** (1997), 206–218.

[Press *et al.* 02] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery, *Numerical recipes in c++*, second ed., Cambridge Press, 2002.

[Ringbeck *et al.* 07] T. Ringbeck, T. Moeller, and B. Hagebeuker, *Multidimensional measurement by using 3-D PMD sensors*, Advances in Radio Science **5** (2007), 135–146.

[Rodehorst & Hellwich 06] Volker Rodehorst and Olaf Hellwich, *Genetic algorithm sample consenus (gasac) - a parallel strategy for robust parameter estimation*, CVPR, 2006.

[Rousseeuw 84] Peter J. Rousseeuw, *Least median of squares regression*, Journal of the American Statistical Association (1984), no. 79, 871–880.

[Sethian 99] James Albert Sethian, *Level set methods and fast marching methods*, Cambridge University Press, 1999.

[Shariat & Price 90] H. Shariat and K.E. Price, *Motion estimation with more than two frames*, PAMI **12** (1990), no. 5, 417–434.

[Shi & Tomasi 94] Jianbo Shi and Carlo Tomasi, *Good features to track*, IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 1994, pp. 593–600.

[Silveira *et al.* 07] Geraldo Silveira, Ezio Malis, and Patrick Rives, *An efficient direct method for improving visual slam*, IEEE International Conference on Robotics and Automation (ICRA), 2007, pp. 4090–4095.

[Soatto & Perona 97] S. Soatto and P. Perona, *Recursive 3-d visual motion estimation using subspace constraints*, IJCV **22** (1997), 235–259.

[Stein 04] Fridtjof Stein, *Efficient computation of optical flow*, Proceedings of the 26th DAGM Pattern Recognition Symposium (Tübingen, Germany), Springer, August 2004.

[Stein *et al.* 00] Gideon Stein, Ofer Mano, and Ammon Shashua, *A robust method for computing vehicle ego-motion*, IEEE Intelligent Vehicles Symposium (IV), 2000.

[Stewart 99] Charles V. Stewart, *Robust parameter estimation in computer vision*, Society for Industrial and Applied Mathematics (SIAM) Review, 1999, pp. 513–537.

[Stewénius *et al.* 06] H. Stewénius, C. Engels, and D. Nistér, *Recent developments on direct relative orientation*, ISPRS Journal of Photogrammetry and Remote Sensing **60** (2006), 284–294.

[Sun *et al.* 06] Zehang Sun, George Bebis, and Ronald Miller, *On-road vehicle detection: A review*, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) (2006), 694–709.

[Thrun *et al.* 05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox, *Probabilistic robotics*, The MIT Press, 2005.

[Tian *et al.* 96] Tina Y. Tian, Carlo Tomasi, and David J. Heeger, *Comparison of approaches to egomotion computation*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996, pp. 315–320.

[Tomasi & Kanade 91] Carlo Tomasi and Takeo Kanade, *Detection and tracking of point features*, Tech. report, 1991, Carnegie Mellon University Technical Report CMU-CS-91-132.

[Torr & Zisserman 00] P. Torr and A. Zisserman, *Mlesac: A new robust estimator with application to estimating image geometry*, Computer Vision and Image Understanding **78** (2000), 138–156.

[Torr 98] P. H. S. Torr, *Geometric motion segmentation and model selection*, Royal Society of London Philosophical Transactions Series A **356** (1998), 1321 – 1340.

[vdHengel *et al.* 07] Anton van den Hengel, Wojciech Chojnacki, and Michael J. Brooks, *Determining the translational speed of a camera from extended optical flow*, Complex Motion, Proc. 1st Int. Workshop (Bernd Jähne, Rudolf Mester, Erhardt Barth, and Hanno Scharr, eds.), Springer, Günzburg, Germany, 2007, pp. 190–197.

[Verri *et al.* 89] A. Verri, F. Girosi, and V. Torre, *Mathematical properties of the two-dimensional motion field: from singular points to motion parameters*, J. Opt. Soc. Am. **6** (1989), 698–712.

[vHelmholtz 25] Herrmann von Helmholtz, *Treatise of physiological optics*, j. p. c. southall ed., Dover, 1925.

[Vidal 05] René Vidal, *Multi-subspace methods for motion segmentation from affine, perspective and central panoramic cameras*, IEEE International Conference on Robotics and Automation (Barcelona, Spain), Springer, 2005.

[Wagner *et al.* 99] Robert Wagner, Feiyu Liu, and Klaus Donner, *Robust motion estimation for calibrated cameras from monocular image sequences*, Computer Vision and Understanding **73** (1999), no. 2, 258–268.

[Welch & Bishop 01] Greg Welch and Gary Bishop, *An introduction to the kalman filter*, ACM SIGGRAPH Course Notes, 2001.

[Woelk & Koch 04] Felix Woelk and Reinhard Koch, *Fast monocular bayesian detection of independently moving objects by a moving observer*, Proceedings of the 26th DAGM Pattern Recognition Symposium (Tübingen, Germany), Springer, 2004.

[Wu *et al.* 95] T.H. Wu, R. Chellappa, and Q.F. Zheng, *Experiments on estimating egomotion and structure parameters using long monocular image sequences*, IJCV **15** (1995), no. 1-2, 77–103.

[WürzWessel 04] Alexander Würz-Wessel, *Free-formed surface mirrors in computer vision systems*, Ph.D. thesis, Eberhard Karls Universitaet Tuebingen, 2004.

[Zhang & Tomasi 02] Tong Zhang and Carlo Tomasi, *On the consistency of instantaneous rigid motion estimation*, International Journal of Computer Vision (IJCV), 2002.

[Zhu *et al.* 05] Juhua Zhu, Ying Zhu, and Visvanathan Ramesh, *Error-metrics for camera ego-motion estimation*, CVPR, 2005.

[Zomotor 91] Adam Zomotor, *Fahrwerktechnik: Fahrverhalten*, second ed., Vogel Buchverlag, 1991.

[Zucchelli *et al.* 02] Marco Zucchelli, Jose Santos-Victor, and Henrik I. Christensen, *Constrained structure and motion estimation from optical flow*, Procs. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2002.

Index