Real Time Motion and Stereo Cues for Active Visual Observers

Mårten Björkman

Stockholm 2002
Doctoral Dissertation
Royal Institute of Technology
Numerical Analysis and Computing Science
Computational Vision and Active Perception Laboratory (CVAP)

Akademisk avhandling som med tillstånd av Kungliga Tekniska Högskolan framlägges till offentlig granskning för avläggande av teknisk doktorsexamen fredagen den 14 juni 2002 kl 10.00 i Sal E1, Kungliga Tekniska Högskolan, Lindstedtsvägen 3, Stockholm.

TRITA–NA–0213
ISSN 0348–2952
ISBN 91–7283–310–6
CVAP265

Abstract

An active visual observer is a vision-guided system that not only reacts to visual stimuli, but is also able to influence the stimuli through its actions. Such a system could for example be an autonomous robot that uses visual information to perform tasks in a dynamic environment. Typical tasks for an autonomous robot may involve fetch-and-carry operations, reconnaissance or delivery missions.

Vision is by an envisaged system used to detect and identify objects, structures and events in the scene. Some of these objects might be familiar, whereas others are new to the observer. Their characteristic properties may therefore not be known a priori. Hence the system needs to utilize any available source of visual information - approaches based on multiple cues are required. The overall goal is to determine both "where" the objects are and "what" they are, i.e. location and identification.

This thesis concentrates on stereo and motion cues. These cues are especially strong since they indicate where objects are located and how they move in relation to the observer. Such information may facilitate extraction of more object-specific data. Since other independent objects may act within the same environment, the observer has to react fast enough in relation to changes of the environment, that is it has to be working in real-time.

The contributions in this thesis may be divided into three themes; evaluation of algorithms and models sufficient for real-time use, development of new efficient methods, and integration of different sources of visual information. The evaluations involve methods for optical flow and binocular stereo estimation, and structure-from-motion. The analyses are done in terms of accuracy as well as speed. New methods are developed for epipolar geometry and ego-motion estimation, image stabilization, and detection of independent motion. Much attention is spent on the efficiency and robustness of these methods.

Towards the end of the thesis different sources of visual information are integrated into a complete system, from which image regions of interest may be automatically obtained, while dynamically changing fixation. The system operates at 9 Hz on a single processor running at 1.2 GHz.

Acknowledgments

A number of people and groups have been actively supporting me during my years working on this thesis. Others have inspired me without necessarily knowing it themselves. These people ought to be recognized. I would thus like to thank the following people:

Jan-Olof för drömmar, visioner och kontinuerligt stöd,
Mor och Far för nyfikenhet och envishet,
Kajsa och Gustav för skratt och support,
Henrik för ambitioner och lösningar,
Axel för mental träning och galna idèer,
Lars P för en måttlig dos spänning i vardagen,
Eric för korrekturläsning och goda idèer,
Dani och Patric för wienerbröd och stöd,
Hanna och Helena för sympati och frukost,
Maria för dans och vänskap,
Emma, Jeanna och Erhan för frågor och förklaringar,
Nordlund, Kourosh,Lars B och Pär för hjälp att komma igång,
Nillius, Micke, Dennis, Marco, Carsten och Martin för gemenskap i arbetet,
Anders, Magnus och Mattias för robotar som jäklas,
Daniel, Stefan och Tony för tips och förslag,
Kraftwerk och DJ Todd för musik och inspiration,
Phenomena och Fairlight för kamratskap och framgång,
Mia för kärlek, stöd och stabilitet
och tack alla dem jag lyckats glömma.

This research has been sponsored by the *Swedish Foundation for Strategic Research* through the Center for Autonomous Systems. The funding is gratefully acknowledged.

Contents

Chapter 1

Introduction

For well over forty years scientists have been striving towards the goal of making computers see. Even if progress has been made, noone has ever come close to what the general public would consider to be a "seeing" system. What was once considered plausible, soon turned out to be much harder than expected. It is not self-evident what an artificial "seeing" system actually is. Vision, as we know from biology, is embodied, which means that there is something that sees. This "something" is involved in a set of tasks and activities, and vision is used to accomplish these. The world in which this occurs is three-dimensional and in it there are other independently acting and moving systems that need to be observed and maybe also interacted with.

A system of the type we envisage could be an autonomous robot using vision to carry out a variety of tasks, such as fetch-and-carry operations, or reconnaissance or delivery missions in environments inaccessible or hazardous to humans. An important aspect of such scenarios is that the environment, like also our everyday environment, is not known in detail and that unexpected events occur in it. To cope with such unpredictability, to detect and identify objects and events in the environment, such a system needs to utilize every possible source of information. For a "seeing" system this implies that different types of visual information should be used and that different algorithms may be appropriate. Another consequence of the sketched scenario is the real-time requirement. The system has to be able to respond to stimuli and events in limited time.

The work described in the thesis should be seen in such a context. Important themes are the integration of multiple types of visual information, the selection between different algorithms and the design of fast algorithms for 3D visual analysis. Vision provides rich information both about "where" things are and "what" they are. This thesis deals mainly with the "where" part, although important input to processes for recognizing "what" are provided through figure-ground segmentation. Before we describe this in more detail we will give an overview of the problems addressed in the following chapters.

1.1 Outline

An active visual system working in a dynamic environment, needs the ability to sample the environment and determine the location and extension of objects present in the scene. It should be able to react to unexpected events, as well as recognize objects that are relevant to the task at hand. This thesis concentrates on the problems of determining the location, extension and relative motion of such objects, without necessarily determining what they look like. We also present a complete system, where the developed methods are integrated.

The system consists of three separate processing paths; the optical flow path, the binocular disparity path and the independent motion path. Each path contributes with different kinds of information derived from the visual input. The paths are not isolated, but share information that is relevant for different paths. In the end of the thesis cues from the different paths will be integrated, so that image regions of interest can be found. From these regions the observer might find an updated gaze direction.

Binocular stereo The location and extension of an object can only be determined, if the image region corresponding to its projection can be segmented out from the visual data. Binocular disparities will be used for this purpose. A number of disparity algorithms will be evaluated for accuracy as well as speed. It will be shown that the difference between different algorithms is primarily in the density of the calculated disparities and not in the accuracy itself.

In order for the observer to relate the disparities to a position in 3D space, the epipolar geometry, that is the relative positions and orientations of the cameras, has to be determined. This information is also essential when the observer wants to change the fixation point through a saccade from one point in 3D space to another. In the context of a binocular stereo head a number of methods for epipolar geometry estimation will be developed. These methods are based on corner features that are extracted from the left and right camera images. Considerable attention is devoted to making the process as robust as possible.

Optical flow Optical flow, that is the temporal changes of image brightness, is needed in order to determine the relative motion of the objects. It may also serve as an additional cue for object segmentation. Objects located close to each other in 3D space can be separated based on their individual motion. Three different optical flow algorithms will be tested for real-time use. Even if the algorithms are implemented as efficiently as possible, only one will turn out to be fast enough.

Since the optical flow algorithms typically produce accurate data only within a limited range of image motion, it is essential that the flow is kept within this range. This means that images have to be stabilized, so that the dominant visual motion is canceled out before optical flow is calculated. This is done by changing reference points in the images, rather than controlling the camera motions. However, it is possible for the observer to use the extracted information for that purpose. Three different stabilization methods are evaluated, out of which one is a new algorithm based on contour points.

Independent motion The optical flow cannot be fully interpreted unless the motion of the observer itself, the ego-motion, is known. Using ego-motion information together with disparities and optical flow, it is possible to determine which parts of the scene belong to a static background and where independently moving objects can be seen. The knowledge of independently moving objects is critical, since such objects might directly affect the observer in its doings. An efficient approach for independent motion detection will be presented. The approach is based on forward-prediction and subtraction of image data.

In an effort to find an appropriate method for ego-motion estimation, six different structure-from-motion algorithms will be analyzed, three linear algorithms and three iterative ones. It will be shown that monocular information is hardly sufficient to accurately and robustly determine ego-motion. However, using information from stereo the problem can be simplified. The result will be sufficient for real-time use, both in terms of robustness and speed.

1.2 Stereo and motion cues

In the thesis numerous visual cues based on binocular stereopsis and motion will be studied in the context of a mobile seeing system capable of controlling gaze. These cues are especially rich in that they include information on how the scene, observed by the cameras, can be segmented into different objects in 3D space. From biology we known that the paths in visual cortex can be divided into "where" and "what" paths. The cues dealt with in this thesis primarily belong to the "where" paths. These paths determine where a distinct 3D object is located, but not necessarily what it looks like. Such information is instead provided by the "what" paths, which include cues from e.g. colour, shape and texture. Since these paths will not be covered in greater detail, the presented system is far from being a complete system. In order for the observer to recognize a familiar object, the "what" paths are essential. However, in order to determine what a particular object looks like, one first has to know where it is located.

The difficulty of dividing an image into different 3D objects is often underestimated. For human beings the process is more or less automatic, since we do not have to spend any effort trying to determine which region of the retinal image belongs to what object. It is not just that this segmentation is done automatically, relevant image regions also tend to "pop out" without conscious effort, directing our gaze towards objects that might be of interest. This process is often called preattentive vision, since no particular attention is required in order for the process to work. The goal of this thesis is to perform the same task computationally. Most effort is spent on the individual cues, and towards the

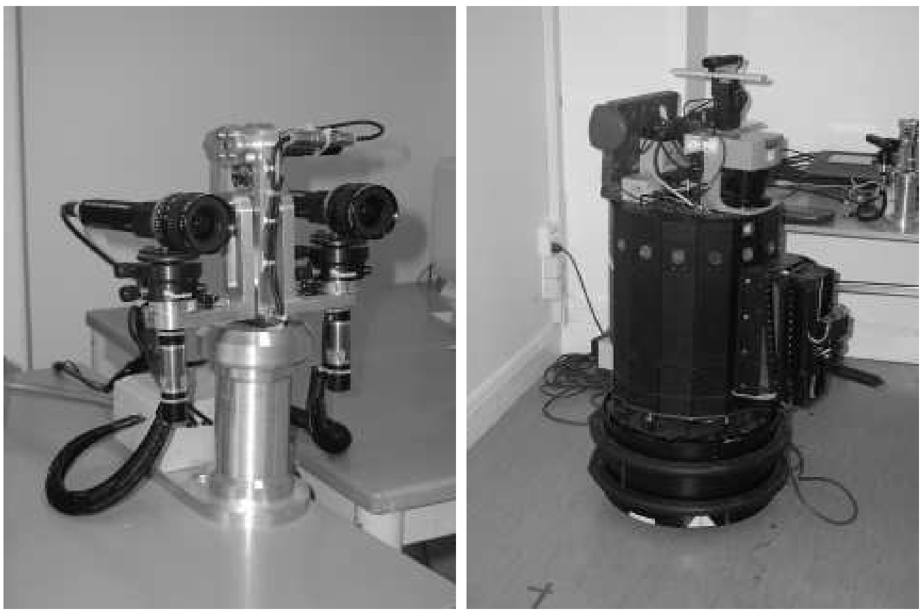

Figure 1.1. A Yorick stereo head (left) and a Nomad 200 platform (right).

end of the thesis a framework for the whole system will be given. The experiments, that will be presented in the following chapters, were performed on the Nomad 200 system (Andersson et al. 1999) shown in the right image of Figure 1.1. A Yorick binocular stereo head (Sharkey et al. 1993) seen to the left was also used in the study.

1.3 Head and eye movements

An observer moving around in a scene is likely to be most successful if it is able to actively control its gaze. If the gaze direction is changed, such that image data are stabilized in the centre of the images, more information on an observed object may be collected as time goes by. Since motion blurring can be avoided, the quality of data extracted will also be better than would otherwise have been the case. Furthermore, in order for the system to produce accurate stereo data, it is essential that the same object is visible in both images. Thus stabilization has to be performed not just for each individual camera, but also binocularly, so that the cameras can be kept in fixation. However, with information available from monocular stabilization, the process of maintaining fixation may be simplified (Pahlavan et al. 1992, Coombs & Brown 1993).

Stabilization of a moving target is known as smooth pursuit. It may be considered as an attentive process, rather than a preattentive one and will thus not be dealt with in greater detail, even if it is an interesting topic in itself.

We will rather focus on saccades, that is the process of rapidly changing gaze direction from one point of interest to another, and especially what information triggers and controls saccades. Saccades have been studied by numerous people in vision science, although computational attempts have been few (Murray et al. 1995). The reason could be that controlling saccades is harder than smooth pursuit. Naively one may think that this is not the case, since saccades are triggered every now and then, whereas smooth pursuit is continuous and relies on series of images being processed at a very high rate. However, when it has been initialized, a great deal is known about the object being stabilized and computations may be concentrated on a small part of the field of view.

On the other hand, when a saccade is triggered not much is known about existing objects. Thus the preattentive process has to work on all possible image regions, out of which one is used to trigger a saccade. Since prior knowledge about available objects is limited, it might not be clear which visual cues to rely on. Computationally one may be forced to consider a whole range of algorithms, where many turn out to be useless. It is also difficult to know how much information one ought to extract before a saccade is issued. If too much time is spent on gathering information, the region of interest might disappear from the field of view. Without enough information, a saccade might be issued towards a region that does not prove to be interesting at all.

In conclusion, the aim is to extract enough relevant information to properly direct gaze towards objects that might be of use or might interfere with the observer in its tasks. Since the system is intended to work in a dynamic constantly changing environment, very little can be taken for granted and the observer has to rely on many different visual cues that each have to be fast enough for it to react in time. Thus the work presented in this thesis may be characterized as "A bag of tricks", an expression that Ramachandran (1985) once used to describe the human vision system.

1.4 Design criteria

The work in this thesis is in a sense biologically inspired, but not necessarily biologically plausible. The goal is not to create a model of any biological system, such as the human visual system, but to provide to an autonomous robot the ability to "see". Even if biological systems are good examples of systems that work, the hardware on which the system runs is totally different. One important difference is the high level of interconnectivity between neurons in a biological system, which is hard to match computationally without seriously restricting the efficiency of calculations. However, a number of lessons can be learned from biology, inspiring the designer of computational systems. Biology might suggest what cues to use and how information is to be extracted from the visual input. The level of interaction between different cues is another interesting topic and where, between low- and high-level vision, these interactions should take place. Unlike most computational systems, biological ones include numerous feedback connections from higher levels back to lower ones. This greatly complicates the analysis of such systems. However, the fact that these connections do exist in biological systems, suggests that they ought to be explored also in computational systems.

All algorithms presented in this thesis are intended to run in real-time using off-the-shelf hardware available today. This real-time requirement is necessary in order to perform closed-loop experiments in which the robot interacts with the environment based on the visual input. What this means in terms of updates per second depends on the task at hand. The more dynamic the environment is, the faster responses have to be. It is often claimed that real-time processing is not really necessary when studying computer vision for robotics. Requiring that everything should work in real-time imposes a constraint on algorithms and methods that may be used. Since future computers are likely to be much faster, work done today might prove to be irrelevant in a few years time. In 1965 Gordon Moore predicted that the complexity of integrated circuits will double every 18 months. In terms of processor speeds, this statement has shown to be remarkably close to the truth, even if recent statistics indicate that 2.5 years might be a more accurate figure. However, memory speed has only doubled every ten years. Thus there is a huge gap between processor and memory speed-ups. With this in mind, it is not at all certain that dramatic speed-ups can be expected, since vision operations are indeed memory critical.

1.5 Contributions

The following chapters represent various aspects of a complete system presented towards the end of the thesis. The chapters are ordered so as to reflect the dependency between different components.

Much work has been spent on analyzing algorithms and methods that have earlier been presented by others. Such evaluations can be found in the chapters on optical flow (Chapter 2), binocular disparity (Chapter 5) and ego-motion (Chapter 6). This has been done in the context of an autonomous system working in an indoor environment, which makes this study different from most other studies. Some suggestions on how to make implementations efficient have also been made, such as the combination of a preconditioned conjugate gradient algorithm with robust M-estimators for optical flow calculation.

An approach for rotational stabilization of two consecutive images is presented in Chapter 3. This method is based on contour points and has the speed and robustness of corner based methods, but is less likely to collapse when the visible scene lacks texture. Another method based on corner features is presented in the same chapter. Chapter 4 contains a large portion of the contributions of the thesis. A binocular stereo head is described in terms of an essential matrix, with the number of free variables kept as few as possible. It is shown how the epipolar geometry can be robustly estimated. An alternative and more efficient iterative method, based on the binocular optical flow constraint, is also presented and evaluated through series of experiments.

A number of ways to combine stereoscopic information with image features matched in time is given in Chapter 7. It is shown how features triangulated using the epipolar geometry may simplify the problem of determining ego-motion. From this information an efficient way of finding independently moving objects in the scene is presented. Different stereo and motion cues are then integrated into a complete system in Chapter 8, with the implementation of each component described in detail. A suggestion is given on how bottom-up cues could be integrated with top-down information. It is also shown how a maximum a posteriori problem may be solved using graph cuts and used for filling-in of information in textureless image regions.

1.6 Published papers

Much of the work presented in this thesis has been published elsewhere and portions of especially Chapters 4, 5, 7 and 8 originate from the following papers.

1. M. Björkman and J-O. Eklundh, 'Real-Time Epipolar Geometry Estimation of Binocular Stereo Heads', *IEEE Trans. Pattern Analysis and Machine Intelligence*, Mar 2002.

2. M. Björkman and J-O. Eklundh, Dynamic Fixation in an Active Visual Agent, *in* 'Robot Vision', Auckland, New Zealand, pp. 1–8, Feb 2001.

3. M. Björkman and J-O. Eklundh, Depth and Motion from a Fixating Binocular System, *in* 'Vision, Modeling, and Visualization', Saarbrücken, Germany, Nov 2000.

4. M. Björkman and J-O. Eklundh, A Real-Time System for Epipolar Geometry and Ego-Motion Estimation, *in* 'IEEE Computer Vision and Pattern Recognition', vol. 2, Hilton Head, SC, pp. 506–513, Jun 2000.

5. M. Björkman and J-O. Eklundh, Real-Time Epipolar Geometry Estimation of Binocular Stereo Heads, Tech. Report ISRN KTH/NA/P-00/09-SE, NADA, Royal Institute of Technology, Mar 2000.

6. M. Björkman and J-O. Eklundh, Real-Time Epipolar Geometry Estimation and Disparity, *in* 'Intl. Conference on Computer Vision', Kerkyra, Greece, pp. 234–141, Sep 1999.

Chapter 2

Optical flow

Optical flow, that is the constant change of image data over time, arises either due to objects moving in the scene or as a consequence of the observer moving itself. Conversely, if the optical flow field is measured, these motions may in principle be determined. Flow due to translational observer motion, which is known as motion parallax, depends on the distance from the observer to objects in 3D space. Thus optical flow may also be used for depth perception, similar to stereo vision. The difference is that displacements are typically smaller and measured between two consecutive images and not between the left and right camera images.

This chapter deals with the computation of dense optical flow field, that is deriving flow at every image point. It can be discussed whether a dense field is really necessary, since information such as ego-motion can also be extracted from a sparser field. The reason for calculating a dense field is that it may be used for segmentation of the image space into regions of different motion. Once a region of interest has been found, attention may be directed towards the region, and more accurate information can be extracted locally. After an introductory background survey, we will evaluate three methods for real-time use. Two of these methods are well-known, but the third one is novel in that a fast Preconditioned Conjugate Gradient method has been combined with a robust M-estimator. It will be shown that the low cost requirement seriously constrains the quality of possible results. However, the results are often good enough for figure-ground segmentation to be successful, especially in conjunction with disparities, which will be shown later on.

2.1 Methods for computing optical flow

Methods for calculating optical flow can be divided into a number of classes, which are sometimes hard to separate, since some methods may in fact consist of ideas originating from different classes (Barron et al. 1994, Beauchemin &

Barron 1995). In this study we have classified methods as being either gradient-based, frequency-based, correlation-based or based on motion models, that may be either local or global. Methods may further be divided into groups depending on whether they use robust statistics or not, or if optical flow is solved in a coarse to fine framework.

Gradient-based methods Most methods are based on the assumption that the brightness of a point moving in space will remain unchanged from one frame to the next, that is the brightness of an image point at position $(x(t), y(t))$ at time t is constant, $I(x(t), y(t), t) = K$. A differentiation yields the well-known brightness constancy constraint, which is known as the (first order) Optical Flow Constraint Equation,

$$I_x u + I_y v + I_t = 0, \tag{2.1}$$

where (u, v) are the x- and y-components of the optical flow and I_x, I_y and I_t are the derivatives of the brightness function.

Since there is only one such constraint and two unknown parameters for each image point, the problem of finding the optical flow is under-constrained. One possibility of overcoming this problem, which is referred to as the aperture problem, is through the assumption that the flow is constant in a neighbourhood around the point under consideration. From this assumption Lucas & Kanade (1981) found the optical flow using least squares over such a neighbourhood. Another way of overcoming the aperture problem is assuming that $\frac{d}{dt}\mathrm{grad}\, I = 0$, which leads to the second order optical flow constraints (Uras et al. 1988, de Micheli et al. 1993),

$$\left\{ \begin{array}{lcl} I_{xx}u + I_{xy}v + I_{xt} & = & 0 \\ I_{yx}u + I_{yy}v + I_{yt} & = & 0 \end{array} \right. \tag{2.2}$$

Now instead of a single constraint, two constraints are available and the optical flow can be found separately for each image point.

Optical flow based on only first order constraints, has often been considered as unreliable. However, it can be shown that results may be easily improved through proper choices of pre-filtering, differentiation, neighbourhood and confidence criteria (Brandt 1997). Since the major source of error is in the calculation of image derivatives (Fermüller et al. 2001), the second order derivatives are typically more sensitive to image noise. To overcome this problem, Tretiak & Pastor (1984) used a combination of both first and second order constraints. A more extreme variant is a method of Weber & Malik (1995), who used a whole series of filters of first and second order derivatives to produce an over-constrained system and total least squares for improved robustness in the computations. Another possibility is first using least median of squares to identify outliers in the data and then use weighted least squares to get a final solution (Bab-Hadiashar & Suter 1998). These outliers are points where either the brightness or the constant optical flow assumption fails. Since errors are squared, these outliers would otherwise easily dominate the result.

A number of proposed methods use an additional smoothing constraint to further increase robustness and density of the calculated flow. Horn & Schunck (1981) performed a global regularization with an isotropic smoothing term

$$E_S = \lambda^2 \int (|\nabla u|^2 + |\nabla v|^2)\, d\mathbf{x} \tag{2.3}$$

together with the brightness constancy constraint

$$E_B = \int (I_x u + I_y v + I_t)^2\, d\mathbf{x}. \tag{2.4}$$

They applied an iterative scheme with a rather slow convergence. However, it is possible to solve the same minimization problem using an incomplete Cholesky preconditioned conjugate gradient algorithm (Lai & Vemuri 1998), that convergences in about 10 iterations, without any increased complexity per iteration. A problem with a smoothing term, such as E_S, is that flow data spread equally in all directions, even across occluding edges. Nagel & Enkelmann (1986) instead used, what they called, an oriented smoothness term that restricted the spread to directions along edges. However, since non-occluding edges are treated no differently from occluding edges, this approach might restrict the smoothing too much. Another possibility is explicitly identifying occluding edges (Thompson et al. 1985) and then only spreading information in the direction of the occluding side. Thompson (1998) outlined a method that, using the normal flow on both sides of a boundary, classifies the two sides into an occluding and occluded one. Projections of flow and edges are performed from frame to frame, to gradually improve the results. Another way of preserving discontinuities in the optical flow is using a more robust smoothing term and minimizing the total energy in a non-linear minimization. It is possible to turn the problem into a convex one through the introduction of dual variables and solve it using linear methods (Deriche et al. 1995).

Frequency-based methods A different family of methods uses spatiotemporal motion energy filters to estimate the optical flow (Adelson & Bergen 1985). Moving image structure results in oriented lines in space-time, where the orientation is defined by the velocity. If a Fourier transformation is performed on Equation 2.1 one will get a new constraint in the frequency domain,

$$\omega_x u + \omega_y v + \omega_t = 0, \tag{2.5}$$

that is (u, v) defines a plane in the space of spatiotemporal frequencies ω_x, ω_y and ω_t. Heeger (1987) used a series of band-pass Gabor filters tuned to different spatial and temporal frequencies to estimate the orientation of this plane and thus the optical flow. Observing that the phase component of the filter outputs is more stable to changes in amplitude, Fleet & Jepson (1990) chose another approach. The output of each filter may be expressed as a complex valued function,

$$R(x,y,t) = \rho(x,y,t)e^{i\phi(x,y,t)}, \tag{2.6}$$

which through differentiation results in a constraint on the phase

$$\phi_x u + \phi_y v + \phi_t = 0. \tag{2.7}$$

This method seems to be the method of choice, in an extensive study by Barron et al. (1994), but there has been a number of reports on its sensitivity to noise (Bober & Kittler 1994, Liu et al. 1997). Even at moderate noise levels of about 5% the phase-based method often results in worse accuracy than far less complex methods, such as the approach of Lucas & Kanade (1981). There are alternatives to the use of spatiotemporal Gabor filters, such as Hermite polynomials, which have been used by Liu et al. (1997), together with a more general model of the optical flow.

Correlation-based methods Correlation-based optical flow methods (Anandan 1989, Singh 1990), that is methods based on correlations of local image neighbourhoods, have come under great criticism and have been considered as less reliable than gradient or frequency based methods. It is primarily a consequence of the fact that correlations tend to make estimates biased towards integer flow values and in most studies approaches are tested towards sequences with ground-truth flow limited to a few pixels in magnitude (Barron et al. 1994). Thus the relative errors can be substantial. Gradient and frequency based methods, on the other hand, have serious problems for larger flows, unless they are used in a hierarchical framework or the images have been significantly blurred in advance. However, this blurring typically results in an over-smoothed estimate of the optical flow.

An approach by Anandan (1989) uses sums of squared differences between local neighbourhoods in a hierarchical framework and distinctness of peaks in correlation space as confidence measures. A smoothness constraint on the optical flow is then imposed using estimates from coarser scales and neighbouring pixels, which results in smoothing in directions of lower confidence. A similar two-stage method was presented by Singh (Singh 1990), who formulated the problem as a statistical combination of flow estimates obtained from two sources, the correlation values and the neighbouring estimates. Each source has its own covariance matrix, and the eigenvalues of the inverted sum of these matrix are used as confidence measure.

Local motion models If a segmentation of the image into regions of different motion is given, the calculated optical flow may be improved using spatial consistency. On the other hand, in order to segment objects along motion discontinuities, the flow has to be known. Thus segmentation and flow estimation are two interrelated problems that are hard to separate. As early as in 1979, Fennema & Thompson (1979) used Hough transforms on the optical flow constraint to perform segmentation, but they never went as far as to exploit the segmentation to improve the optical flow. Adiv (1985*a*) instead performed segmentation after estimating the optical flow in order to fit flow data into different regions of planar motion. Bergen, Burt, Hingorani & Peleg (1992) were able to find the flows of two different motion components without an initial segmentation being required. In an iterative scheme using warping and subtraction, the two components were alternately solved for, assuming one of the components to be given.

Since segmentation is such a difficult problem, one could instead use robust methods near motion boundaries. Neighbouring pixels belonging to different moving objects can be treated as outliers, as well as pixels for which the brightness constraint does not hold. For this purpose Black & Anandan (Black & Anandan 1993, Black 1994) used an M-estimator and the Graduated Non-Convexity method of Blake & Zisserman (1987) to iteratively locate a globally optimal solution. Ju et al. (1996) used a similar method with locally affine patches, rather than the optical flow constraint pixel-wise and got exceptionally good results on the well-known Yosemite sequence. Another example of robust methods used for optical flow calculation is an approach of Ong & Spann (1999), who used robust least median of squares regression of affine optical flow models on overlapping blocks. It is also possible to combine robust methods with a hierarchical structure of parametric models (Szeliski & Shum 1996, Memin & Perez 2002), for improved speed and disparity range. However, such models typically have difficulties aligning block edges to occlusion boundaries.

A popular notion in image motion analysis and especially in video coding, is that of scenes segmented into layers. Each layer represents a region of some motion and each pixel is assigned to one such layer. Darrell & Pentland (1991) used support maps to represent the segmentation and the Minimum Description Length principle to determine the preferred number of support maps. Each support map determines the relative weight given to a particular motion hypothesis for each pixel. A similar approach was presented by Ayer & Sawhney (1995), but they used Expectation-Maximization to facilitate a binary assignment of pixels to segments. Initial hypotheses of affine motion can be calculated locally, as done by Wang & Adelson (Wang & Adelson 1993, Wang & Adelson 1994), who then merge similar hypotheses using k-means clustering.

Black & Jepson (Black & Jepson 1994, Black & Jepson 1996) instead used intensity based segmentation of images into regions and fitted these to variable order parametric models, with local deformations near occlusion edges and areas where a planarity assumption does not hold. An approach by Weiss & Adelson (1996) is similar to that of Black & Jepson's in the sense that image intensities are used for segmentation, but this is done globally with the number of layers kept low, reducing the risk of over-segmentation. Unfortunately, planar regions that satisfy an affine motion model are often hard to find and sometimes lead to such over-segmentation. Weber & Malik (1997) instead segmented image sequences into different moving regions using a model based on the affine epipolar constraint and updated these models using Kalman filters.

Global motion models For many applications, such as navigation and surveillance, there is no reason why the optical flow field has to be calculated locally. Assuming that there is a dominant flow component induced by the motion of the observer, it is possible to perform an image registration on the whole image (Bergen & Adelson 1987, Bergen, Anandan, Hanna & Hingorani 1992) and indirectly recover the flow field. In a sense, using one single model covering the whole visual field, is an extreme on a spectrum of approaches, whereas methods using only local information, such as the method of Weber and Malik (Weber & Malik 1995), are located at the other side of the spectrum. Unfortunately, without knowing the depths for every single image point, an image stabilization might result in a residual. However, one might exploit this residual and find the translational direction of the observer (Irani et al. 1994*b*), as the registration eliminates the rotational component. Irani et al. (1994*a*) also tried to find, and as new frames become available update, connected components in the residual, resulting from independently moving objects in the scene. It is further possible to take advantage of multiple frames, using the fact that the flow is embedded in a lower dimensional linear subspace if the scene is rigid (Irani 1999) or planar (Zelnik-Manor & Irani 2000).

2.2 Towards real-time optical flow

If one would like to calculate a dense optical flow field in real-time, what kind of methods should be considered? Since most methods rely on the flow being limited to a few pixels in magnitude, it is essential that the computations can be performed at a high enough frame-rate. The changes from update to update will also be smaller at a higher rate and it is then easier to take advantage of temporal consistency. Thus an “accurate” method running at 1 Hz is not necessarily better than a less accurate one running at 25 Hz, since the changes in the scene during that second might be more than the system can handle. Few attempts have been made to find faster optical flow methods, even if some exceptions do exist (Liu et al. 1997, Camus 1997). The problem of speed is often ignored, with the belief that enough computational power will become available in the future. However, the most accurate methods are many orders of magnitude slower than the fastest ones of today and since higher resolution and better frame-rate might be of more interest than pure accuracy, there will always be a trade-off between computational cost and accuracy.

There is another requirement that limits the number of possible methods. In the case of an active observer it is essential that the latency between an event occurring in the scene and the resulting change being observed in the optical flow is low. This means that calculations should be performed using as few images as possible, preferably only two or three image frames. Typically an odd number of frames is considered, with the calculated flow representing the image velocities of the centre frame. Even if it is possible to use only two frames, it might sometimes be beneficial to use an additional frame, so that the calculated flow actually corresponds to what happens at an event for which an image is available and not an event between two frames.

In this study we have evaluated three different methods that are all based on image gradients. The x-, y- and t-wise derivatives are approximated using three different separable $5 \times 5 \times 3$ kernels. Each kernel consists of a high-pass filter along one dimension and low-pass filters for the remaining two. The high-pass filters $H_x = H_y = \frac{1}{12}(1, -8, 0, 8, -1)$ and $H_t = \frac{1}{2}(-1, 0, 1)$ used for space and time derivatives, are the discrete approximations closest to a continuous differentiation filter. Low-pass filtering is performed with ordinary Gaussian filters, that is $L_x = L_y = \frac{1}{12}(1, 4, 6, 4, 1)$ in space and $L_t = \frac{1}{4}(1, 2, 1)$ in time. Thus with I_3 denoting three consecutive images, the gradients are calculated as follows:

$$\begin{aligned} I_x &= H_x * L_y * L_t * I_3 \\ I_y &= L_x * H_y * L_t * I_3 \\ I_t &= L_x * L_y * H_t * I_3 \end{aligned} \tag{2.8}$$

2.2.1 Weighted least squares

The first implemented method is that of (Lucas & Kanade 1981), which has often shown to be remarkably accurate (Barron et al. 1994) despite its simplicity. Optical flow is estimated through weighted least squares regression on the optical flow constraint given in Equation 2.1, which is performed locally with one window for each image point. That is, if p is a point of interest and N_p its local neighbourhood, the corresponding flow vector (u_p, v_p) is sought such that

$$\sum_{s \in N_p} w_s (I_{x,s} u + I_{y,s} v + I_{t,s})^2, \tag{2.9}$$

is minimized. The window function w_s represents the weights used for each flow constraint, putting more emphasize on the central constraints, than the peripheral ones. A Gaussian filter kernel of dimension 5×5, as given by L_x and L_y above, is chosen for this purpose. Thus an estimate of the optical flow is given by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{A}_p^{-1} \mathbf{b}_p, \text{ where} \tag{2.10}$$

$$\mathbf{A}_p = \sum_{s \in N_p} w_s \begin{pmatrix} I_{x,s}^2 & I_{x,s} I_{y,s} \\ I_{y,s} I_{x,s} & I_{y,s}^2 \end{pmatrix} \text{ and} \tag{2.11}$$

$$\mathbf{b}_p = -\sum_{s \in N_p} w_s \begin{pmatrix} I_{x,s} I_{t,s} \\ I_{y,s} I_{t,s} \end{pmatrix}. \tag{2.12}$$

Problems occur if the second moment matrix $\mathbf{A}_p$ is singular. With line features, which contain no structure in the direction along the line, the least eigenvalue λ_2 of $\mathbf{A}_p$ will be zero, and the matrix cannot be inverted. Then the best one can do is to accept the normal flow

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{\mathbf{b}_p}{trace(\mathbf{A}_p)}, \tag{2.13}$$

that is the optical flow in the direction of the maximum gradient. For uniformly shaded image regions, the largest eigenvalue will also be zero. In order to determine whether the calculated data are to be trusted, some kind of confidence measure is required. The trace of $\mathbf{A}_p$ is one such possible measure (Simoncelli et al. 1991). However, the trace does not discriminate between line structures and more reliable two-dimensional structures that do not suffer from the aperture problem. In this study, like in many other studies, we use the smallest eigenvalue, which can be written more explicitly as

$$\lambda_2 = trace(\mathbf{A}_p)/2 - \sqrt{trace(\mathbf{A}_p)^2/4 - det(\mathbf{A}_p)}. \tag{2.14}$$

For each image position, the corresponding confidence value is stored together with the optical flow estimate. Thresholding is delayed until later, which means that different thresholds, typically between 0.5 and 5, can be used for different operations on the flow. In fact, which measure to use depends on the application. In the case of observer motion and structure estimation, a mixture of optical flow and normal flow might lead to erroneous results. However, if the flow is used in order to judge whether it is compatible with a known motion, normal flow might suffice. The method has been implemented in two different versions, with different ways of representing temporary buffers, as described in the Appendix. Calculating a 192×144 pixel flow field requires about 24.4 ms with gradients stored as whole images, and 12.5 ms if the gradients are calculated on the fly and just stored temporarily.

2.2.2 Fast regularization with smoothing

Another method to be tested is the classical approach of Horn & Schunck (1981), but with the exception that the system is solved using a preconditioned Conjugate Gradient method, with an incomplete Cholesky decomposition as preconditioner (Golub & Van Loan 1996), along the lines described in (Lai & Vemuri 1998). The error function given by Horn & Schunk consists of two parts, one due to the brightness constancy assumption and another for smoothness, that is

$$E = \sum_p (I_{x,p}u_p + I_{y,p}v_p + I_{t,p})^2 + \lambda(|\nabla u_p|^2 + |\nabla v_p|^2), \tag{2.15}$$

with λ used as the relative weight between the two constraints. If the gradient of the x-wise flow component at a point $p = (x, y)$ is approximated by $\nabla u(x, y) \simeq$

$[u(x+1,y)-u(x-1,y), u(x,y+1)-u(x,y-1)]^\top$, the corresponding factor of the smoothing constraint can be written as

$$\sum_p |\nabla u_p|^2 \simeq \sum_{x,y} u(x,y)\Delta u(x,y), \quad \text{where} \tag{2.16}$$

$$\Delta u(x,y) = 4u(x,y) - u(x+1,y) - u(x-1,y) - u(x,y+1) - u(x,y-1)).$$

The y-wise part of the smoothing constraint can be approximated in a similar manner. With all flow vectors stacked on top of each other, such that the first half consists of the u components and the bottom half contains the v components, a $2N$-vector $\mathbf{u}$ is created, where N is the total number of image points. The energy to be minimized is then given by

$$E(\mathbf{u}) = \mathbf{u}^\top \mathbf{A}\mathbf{u} + \mathbf{b}^\top \mathbf{u} + \sum_p I_{t,p}^2, \tag{2.17}$$

with $\mathbf{b}$ denoting a $2N$-vector with $I_{x,p}I_{t,p}$ and $I_{y,p}I_{t,p}$ as elements. The $2N \times 2N$-matrix $\mathbf{A}$ includes 7 diagonals and can be written as four submatrices,

$$\mathbf{A} = \left(\begin{array}{c|c} \mathbf{A}_{xx} + \lambda \mathbf{A}_s & \mathbf{A}_{xy} \\ \hline \mathbf{A}_{xy} & \mathbf{A}_{yy} + \lambda \mathbf{A}_s \end{array} \right), \tag{2.18}$$

such that $\mathbf{A}_{ij}$ are diagonal matrices with $I_{i,p}I_{j,p}$ as elements and $\mathbf{A}_s$ is given by the $\Delta u(x,y)$'s in Equation 2.16.

With the Conjugate Gradient method a solution to a symmetric positive definite problem is found iteratively, through gradient descents in directions $\mathbf{p}_k$ $\mathbf{A}$-orthogonal to previously explored directions, that is $\mathbf{p}_k^\top \mathbf{A}\mathbf{p}_j, 0 \leq j < k$. Such a direction can be found as a linear combination of the previous direction $\mathbf{p}_{k-1}$ and the current residual $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{u}_k$. If the matrix $\mathbf{A}$ is ill-conditioned, the convergence rate will be weak and a so called preconditioner can be used instead. A preconditioner is chosen such that it improves the condition number of the system and at the same time is a good approximation of $\mathbf{A}$.

With a sparse matrix $\mathbf{A}$, such as in our problem, an incomplete Cholesky decomposition may serve as a preconditioner. The benefit of using incomplete Cholesky factorization, instead of standard Cholesky factorization, is because the sparsity can be kept, which is necessary due to the large size of $\mathbf{A}$. In conclusion, the preconditioner used here is a matrix $\mathbf{P} = \mathbf{L}\mathbf{L}^\top$, such that $\mathbf{L}$ is a sparse lower triangular matrix and $\mathbf{P}$ is close to $\mathbf{A}$. In this study an $\mathbf{L}$ matrix, with 8 diagonals is used, following the derivations in (Lai & Vemuri 1998). Because of its simple structure, $\mathbf{P}\mathbf{z}_k = \mathbf{r}_k$ can easily be solved, as seen in the algorithm of Figure 2.1.

With a smoothing factor λ of about 2, the method converges in approximately 10 iterations, which is two orders of magnitudes faster than the method proposed by Horn & Schunk. Its major weakness is the fact that 20 floating point values have to be stored and updated for each image point. On our machine the time required for convergence is 80 ms for a flow field of dimension 96×72. It is worth pointing out that the initial flow $\mathbf{u}_0$ in Step 1 of the algorithm may originate from a coarser scale flow field or from a previous instance in time, resulting in an even faster convergence.

Preconditioned Conjugate Gradient

```
1.  Choose $\mathbf{u}_0$
2.  $k = 0, \ \mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{u}_0$
3.  Solve $\mathbf{P}\mathbf{z}_0 = \mathbf{r}_0, \ \mathbf{p}_0 = \mathbf{z}_0, \ \gamma_0 = \mathbf{r}_0^\top \mathbf{z}_0$
4.      $\alpha_k = -\gamma_k / \mathbf{p}_k^\top \mathbf{A}\mathbf{p}_k$
5.      $\mathbf{u}_{k+1} = \mathbf{u}_k - \alpha_k \mathbf{p}_k$
6.      $\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{A}\mathbf{p}_k$
7.      Solve $\mathbf{P}\mathbf{z}_{k+1} = \mathbf{r}_{k+1}$
8.      $\gamma_{k+1} = \mathbf{r}_{k+1}^\top \mathbf{z}_{k+1}$
9.      $\beta_k = \gamma_{k+1}/\gamma_k$
10.     $\mathbf{p}_{k+1} = \mathbf{z}_{k+1} + \beta_k \mathbf{p}_k$
11.     $k = k + 1$
12. Until convergence, return to 4.
```

Figure 2.1. Preconditioned Conjugate Gradient method

2.2.3 Making the constraints robust

Unfortunately, the method of Horn & Schunk uses a smoothing constraint that is applied globally, without taking possible occlusion boundaries into consideration. The resulting optical flow tends to be over-smoothed, which makes segmentation based on flow difficult. In order to allow discontinuities in the flow field, the quadratic smoothing term ought to be changed to a more robust error function. An estimator being robust means that the influence of a single datum should be bounded as the error increases. With a quadratic function, the influence due to errors grows linearly, which means that the end result will be totally dominated by outliers if such exist. Indeed, occlusion boundaries appear frequently in most image sequences.

As mentioned in the beginning of this chapter, numerous researchers have previously been using robust estimators for optical flow estimation. In this section robust error functions will be used for the brightness as well as smoothing constraint. The method is much similar to one presented by Black (1992) in his thesis, but is different in its implementation. The resulting non-linear optimization problem will be solved iteratively using the Conjugate Gradient method described in Section 2.2.2. The error to be minimized can be expressed as follows:

$$E = \sum \rho_B(I_x u + I_y v + I_t, \sigma_B) + \lambda \, \rho_S(\| \nabla \, [u, v] \, \|, \sigma_S), \tag{2.19}$$

where $\rho_B(x, \sigma_B)$ and $\rho_S(x, \sigma_S)$ are the error functions associated will the brightness and smoothing assumptions. In the implementation from which results will be presented, Cauchy functions,

$$\rho(x, \sigma) = \log(1 + \frac{1}{2}(x/\sigma)^2), \tag{2.20}$$

also known as Lorentzian functions, are used for both error terms. Like other M-estimators, the optimization problem can be solved using iterated reweighted least-squares, with the corresponding weights defined by

$$w(x, \sigma) = \frac{1}{x}\rho_x(x, \sigma) = \frac{1}{\sigma^2 + x^2/2}. \tag{2.21}$$

More explicitly, with ϵ_B and ϵ_S denoting the errors of the brightness and smoothness constraints, the function to be minimized can be written as

$$\begin{aligned} E &= \textstyle\sum_p \omega_B(\epsilon_{B,p})\ \epsilon_{B,p}^2 + \lambda\ \omega_S(\epsilon_{S,p})\ \epsilon_{S,p}^2, \\ \epsilon_{B,p} &= I_{x,p}u_p + I_{y,p}v_p + I_{t,p} \text{ and} \\ \epsilon_{S,p} &= \sqrt{u_{x,p}^2 + u_{y,p}^2 + v_{x,p}^2 + v_{y,p}^2}. \end{aligned} \tag{2.22}$$

The weight functions are evaluated using the errors from the previous pass of an iterative procedure, thus resulting in a series of quadratic problems. Each consecutive quadratic problem is then solved using the Conjugate Gradient method, which in itself is iterative. This means that the whole non-linear optimization problem will be solved using two nested iterative processes. The continuation parameters σ_B and σ_S are decreased for each pass, such that the influence of outliers is gradually reduced. The shape of error and weight functions due to the brightness constraint can be seen in Figure 2.2 and the complete method is summarized as pseudo-code in Figure 2.3.

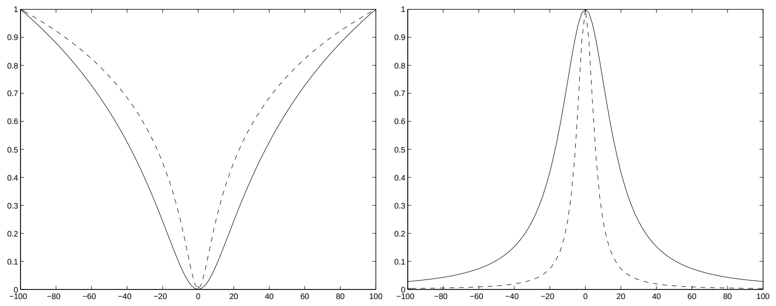

Figure 2.2. The error (left) and weight (right) functions of the Lorentzian estimator used for brightness constancy errors. The solid lines show the initial functions, while the dashed lines show the final ones. In order to facilitate comparison, the graphs have been rescaled so that their peaks coincide.

In the current implementation the continuation parameters are decreased by a factor of 0.5 for each pass, starting at $\sigma_B = 12$ and $\sigma_S = 0.2$. As in the previously presented method the parameter λ, which is fixed to 0.1, denotes the relative importance between the two constraints. However, taking the weight functions into consideration, the difference is in fact $\lambda(\sigma_B/\sigma_S)^2 = 360$ if there are no errors in either constraint. This means that for areas where discontinuities in the flow do not exist, the optical flow due to the robust version is smoothed

Regularization with robust estimators

```
1. Choose initial $\mathbf{u}$, $\sigma_B$ and $\sigma_S$
2.     Warp images using $\mathbf{u}$ and calculate gradients
3.     Calculate weights $\omega_B, \omega_S$
4.     Create preconditioner factor $\mathbf{L}$ (see Section 2.2.2)
5.     Update $\mathbf{u}$ using Conjugate Gradient (see Figure 2.1)
6.     Decrease $\sigma_B$ and $\sigma_S$
7. Until convergence, return to 2.
```

Figure 2.3. Regularization with Cauchy functions

more than in the original method of Section 2.2.2. Experiments indicate that three passes of five iterations each yield results sufficient for our applications, as will be shown in the next section. The computational cost is about 150 ms on a 1.2 GHz Athlon MP machine for flow fields of dimensions 96×72, without taking advantage of flow calculated in previous frames.

2.3 Experiments

In this section a number of experiments will be presented, based on the methods of Lucas & Kanade, Horn & Schunk and robust Horn & Schunk, as presented in Section 2.2. Since our major concern is whether dense optical flow can be used to reliably indicate significant events occurring in the scene, the results will be compared qualitatively rather than quantitatively. Even if segmentation based on optical flow is interesting in itself, it is believed that coarse estimates of independently moving regions in terms of location, heading and size will provide enough information for the system to redirect attention. Once an object is attended to more accurate processes can be applied locally.

A three level Gaussian pyramid of images was used, with data being low-pass filtered and subsampled between each level. The size of the finest scale image was set to 192×144 pixels. Optical flow was then calculated from top to bottom, warping the results from one level to the next, where it is used as initial flow estimates. The reported results of the regularization based approaches, those based on Horn & Schunk, originate from the second finest scale, which is 96×72 in size, in order to match the computational costs reported in Section 2.2. A series of four different experiments were performed to evaluate the three presented methods to find out if they are suitable for real-time use.

Experiment 1

In the first experiment we investigate the general characteristics of the different methods, using a three image sequence of a square translating on-top of a checkerboard background, with the central image shown to the upper-left in Figure 2.4. The results of Lucas & Kanade can be seen in the upper-right image. Due to the fact that this method uses local windows of optical flow constraints, the measures around the boarders of the image are not reliable and have thus been excluded. Since optimization is performed locally, unlike the other methods that also include a smoothing constraint, it does not get any support from more distant regions, which means that sporadic errors often appear in the resulting flow. This can be seen along the upper and left edges of the square, where image structures related to foreground and background are confused.

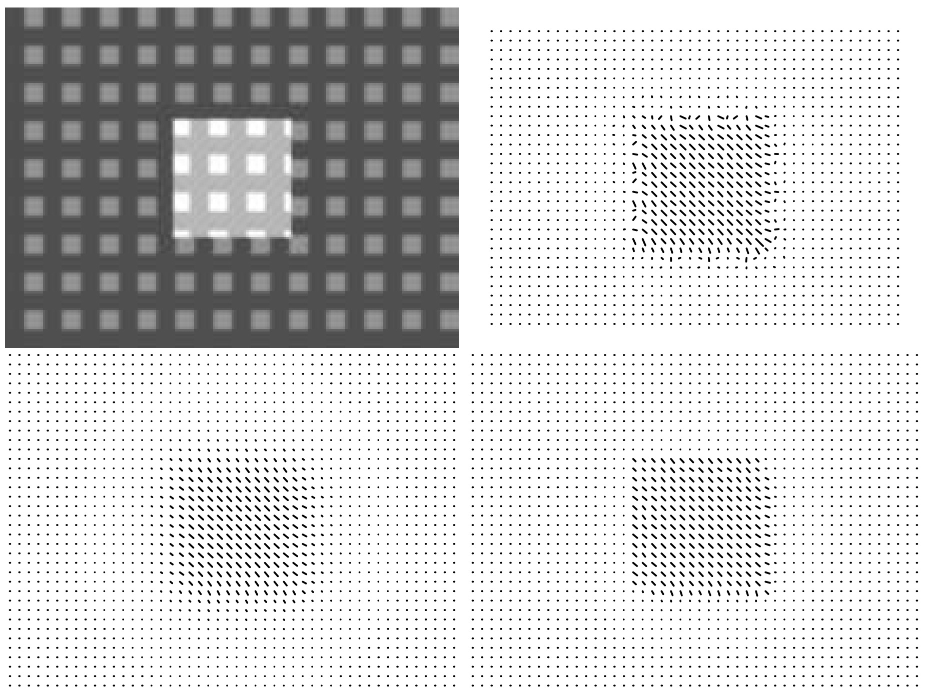
Figure 2.4. The upper-left image shows a square translating one pixel to the right and down. Results from Lucas & Kanade, Horn & Schunk and robust Horn & Schunk are shown in the upper-right, lower-left and lower-right figures respectively.

The method of Horn & Schunk does not suffer from the same sporadic errors, but instead of really resolving the problem, errors are smoothed out. This leads to new errors, when flow is erroneously spread across discontinuities. A method that does not suffer from these two kinds of errors, at least not in the basic example presented here, is the robust version of Horn & Schunk's method. It is worth pointing out that discontinuities in the calculated flow not necessarily correspond to the exact locations of occlusion boundaries in the original sequence. Structure is still needed on both sides of a boundary, in order for the method to identify it as a discontinuity. Its estimated location will be somewhere in between image structures on either side. Thus the moving square in the lower-right flow diagram of Figure 2.4 is slightly larger than in the stimulus.

Experiment 2

The next sequence, of which three images are shown in the left column of Figure 2.5, is taken by an autonomous robot moving around a living-room. The translational motion is along the optical axis with a speed of about 4 cm per update or 1 m/s when operating at 25 Hz. The calculated optical flow can be seen in the centre and right columns of the same figure and gives an indication of what can be expected from many everyday scenes. Flow in the x-wise direction corresponds to the central columns, whereas the right column shows the y-wise component. The nearest object is the chair on the left, located about 2 m away from the viewer. It is here the maximum optical flow can be found, with a peak of about 2 pixels downwards, which is shown as the lighter areas of the right column. As can be seen from the results, the flow is expantional, which is expected from translational forward motion.

The darker regions of the upper row indicate areas of low confidence when calculating flow using Lucas & Kanade's method, that is areas where the least eigenvalue of the second moment matrix $\mathbf{A_p}$ is close to zero. An expanding motion field is evident in these images, but the flow is not accurate enough for the chair and wheelchair to be seen. Some obvious errors do exist, especially where the background can be seen through the back of the chair. If the magnitudes of flow vectors are analyzed in greater detail, many vectors prove to be underestimated, a consequence of the bias towards zero in the least squares optimization, as explained by Fermüller et al. (2001).

In the flow due to Horn & Schunk, which can be seen in the second row, regions that represent the left chair and wheelchair can vaguely be seen, but due to extensive smoothing, it is questionable if it can be used for segmentation. In a sense this is understandable since the magnitude of flow induced by forward observer motion is typically quite small, especially near the focus of expansion. The same thing applies to the robust version, even if some discontinuities are visible, such as the seat of the chair. Contrary to what is expected, flow may in fact float across discontinuities, which is partially due to the fact that the filter kernels have a certain width. It might also be the case that some discontinuities only exist at finer scales and that smoothed initial data are pushed downwards from coarser levels, where these discontinuities have not yet become visible. Using more iterations than those suggested in Section 2.2, seems to support this notion. It is also possible to use lower values of the smoothing parameter λ at coarser scales, instead of the same value for all scales, which was the case in this experiment.

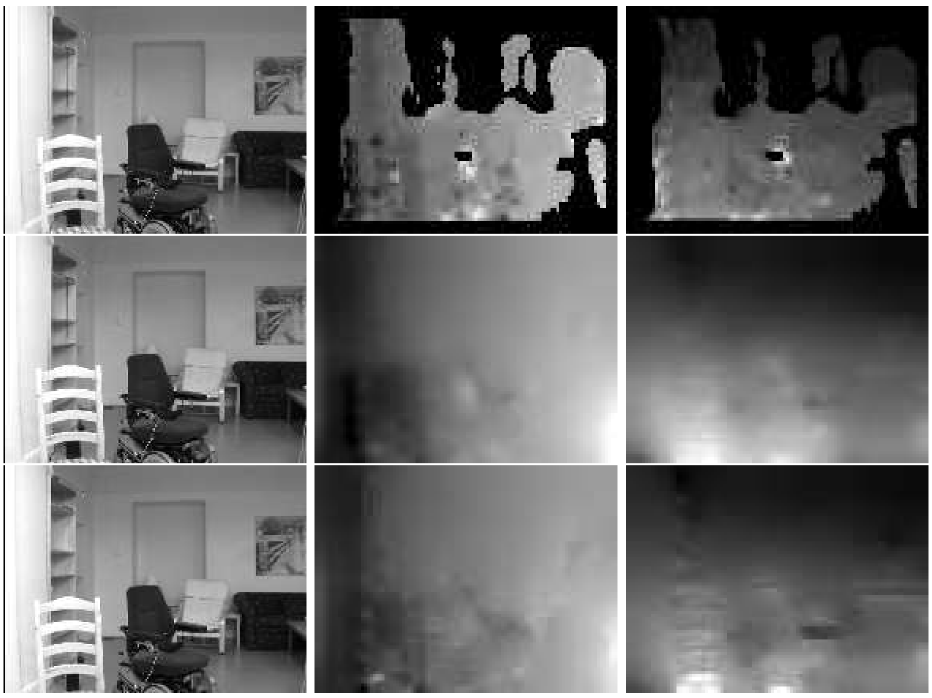
Figure 2.5. The left column shows three images from a typical sequence. The u and v flow components are given in the centre and right columns. Results from Lucas & Kanade, Horn & Schunk and robust Horn & Schunk are in the upper, middle and lower rows. Regions without enough texture are shown as dark areas.

Experiment 3

In this experiment shown in Figure 2.6, the conditions are similar to the previous one, except for the fact that the wheelchair is controlled by a person, who is driving it at a speed of about 3.5 m/s to the right. This results in a maximum flow of 11 pixels to the right and 3 pixels down in an image of dimensions 192×144. This is quite a lot compared to typical testcases presented elsewhere, for example in the work of Barron et al. (1994). This might seem an extreme case, but it does occur frequently in typical indoor scenes. What is quite striking is that Lucas & Kanade's algorithm does considerably well, compared to, for example, the original version of Horn & Schunk. Quantitatively, the results are rather bad for the region representing the moving wheelchair, but it is quite clear that it can indeed be used for segmentation. The results are bad because the magnitude is grossly underestimated. The reason is that even at the coarsest scale the flow is greater than what the method reliably can estimate. Erroneous data survive from level to level, without being significantly improved. Normally the estimate of flow is improved at each new level, but that requires the initial error to be limited to about 1.5 pixels. However, even if the results are erroneous, they

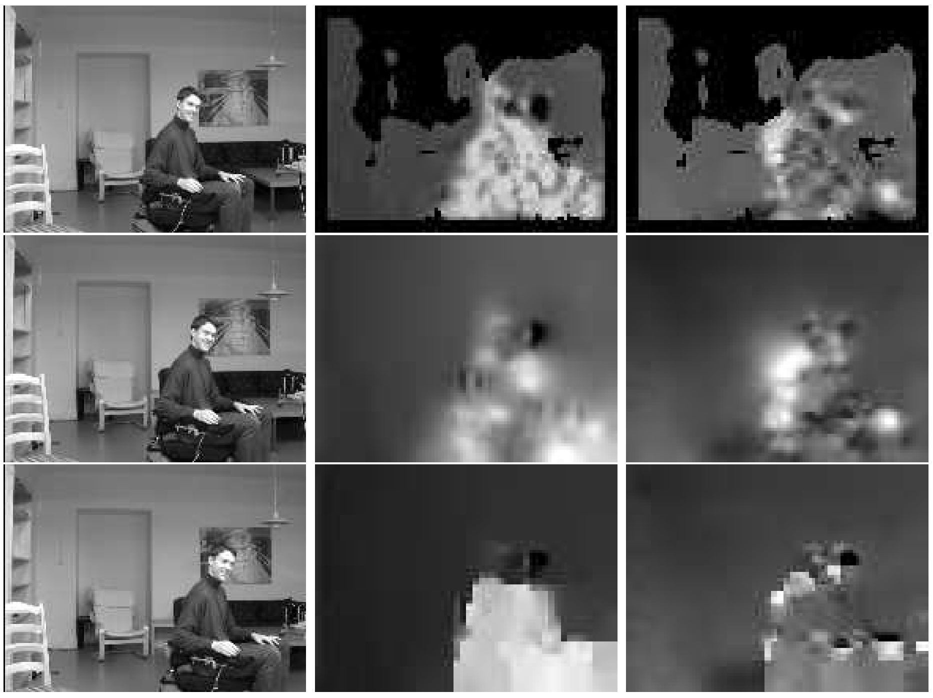

Figure 2.6. The left column shows three images from a typical sequence. The u and v flow components are given in the centre and right columns. Results from Lucas & Kanade, Horn & Schunk and robust Horn & Schunk are in the upper, middle and lower rows. Regions without enough texture are shown as dark areas.

capture the principal behaviour of the motion. The y-wise component of the right columns seem to be recovered worse than the x-wise component, but that is primarily due to the scaling of the intensities in the presentation.

The method of Horn & Schunk is on the other hand forced to comply with the smoothing constraint, which results in complete failure. The only thing one can be certain of is that something is moving and that it is probably moving to the right. If the method is instead made robust, the calculated flow looks much more promising. It is still underestimated, but not as much as by Lucas & Kanade's method, and with additional iterations the result will gradually improve, although more and more slowly. The occluded region in front of the wheelchair does not contain enough structure to prevent it from being incorporated into the foreground object. Much like Lucas & Kanade's this approach encounters problems already at the coarsest scale, since the flow is close to the limit of what can be estimated. Unless convergence occurs at the first iteration, it will most likely never converge. This is what happens on the upper part of the object, with the unfortunate consequence of the disabled person in the wheelchair being beheaded. Since the kernels used for gradient estimation stretch across occluding edges, the results will here be too erroneous for the method to produce good enough initial estimates.

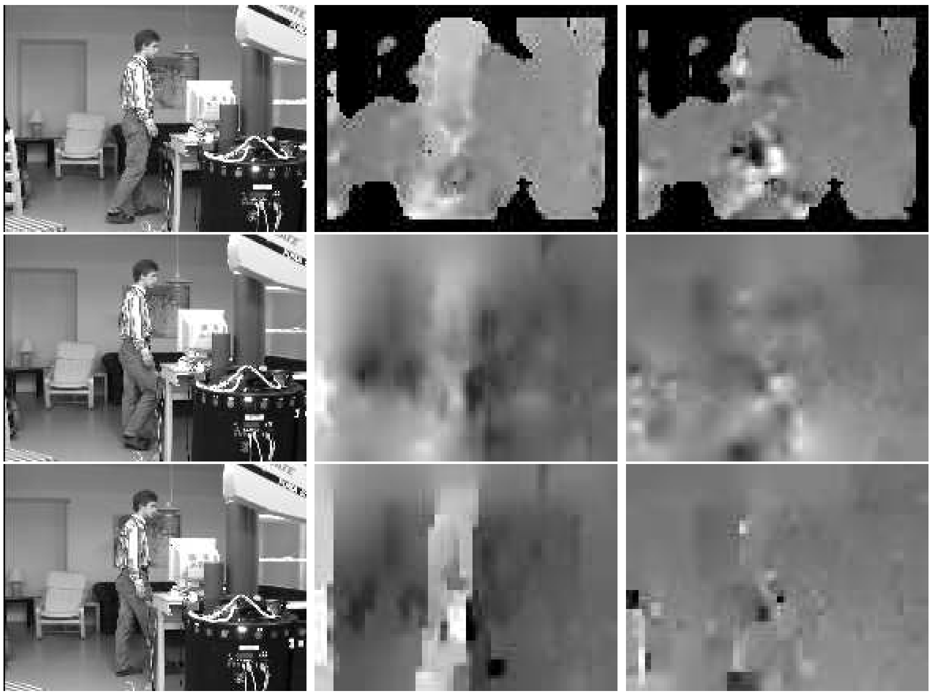

Figure 2.7. The left column shows three images from a typical sequence. The u and v flow components are given in the centre and right columns. Results from Lucas & Kanade, Horn & Schunk and robust Horn & Schunk are in the upper, middle and lower rows. Regions without enough texture are shown as dark areas.

Experiment 4

The last experiment is also the most challenging one, in that the observer rotates at a speed of 35°/s, while an independently moving object generates deformable optical flow. Like the previous example, this case is not extreme either, since the walking person seen in Figure 2.7 is only moving at about 3 m/s. This results in the background moving to the left at a speed of more than 10 pixels per update. Thus the conditions are in a sense similar to the previous case, but with the distinction that the foreground is static, while the background is rotating and not the other way around.

It is quite clear from the results that this is indeed a harder problem. With the robust version of Horn & Schunck's approach, the foreground object may still be segmented from the background. This is likely to be the case also for the method by Lucas & Kanade, even if the borders are less distinct. Because of the size of filter kernels and due to the fact that optimization is performed in windows of 5×5 pixels, the region in foreground has increased in width compared to the stimulus. That was the case also in earlier examples, but it is more evident here. In fact, the resulting flow is spread out, even though there ought to be sufficient data in the background. From Equation 2.12 in Section 2.2.1, one may conclude that if either the spatial or the temporal gradient is zero, the product will also be zero. This might occur when image data have not been properly warped, due to errors in the previous coarser level. As a result the constraint is changed, such that zero flow is favoured, with the consequence that flow vectors tend to be spread towards the side of largest flow, where such errors are most frequent.

2.4 Conclusions

In this chapter three methods for calculating optical flow on sequences of images have been explored, with the aim of finding a suitable approach for a real-time active vision system. The intention is to extract enough information to judge whether anything of significance occurs in the scene. The belief is that coarse estimates of motion, location and size, derived from the optical flow, can be used to attract the attention of the observer, as will be described in Chapter 8. Once an object has been attended to, more accurate information can be calculated locally, in order to determine properties such as shape.

From the experiments one may conclude that a smoothing constraint is indeed useful, but only in conjunction with a robust estimator. However, it is far from certain that such a method would be better than using only the optical flow constraint, such as in the method by Lucas & Kanade. The one order of magnitude difference in computational cost is such that a simpler method might enable a far greater update frequency, which then makes the optical flow estimation problem easier to solve. One major obstacle is the fact that the flow field can be grossly underestimated and data be spread across occlusion boundaries. In the next chapter we will try to stabilize the background (or foreground) in order to compensate for this weakness.

Chapter 3

Stabilization

Calculating optical flow, as was done in the Chapter 2, is in a sense a spatial matching operation. Matching is easier if one knows the expected range at which one should search for matches. Similar matching operations may occur when estimating binocular disparities or when the incoming stimulus are compared to a previously stored memory map. In all these cases, a system would benefit from a coarse approximation of the appropriate matching range. In this chapter we will determine this range for matching between two consecutive images. We call this operation stabilization. A new method based on contour points will be presented and evaluated against two other methods, one based on corner features and another one that uses all image points. The evaluation is done in terms of accuracy as well as speed.

Stabilization may also be used to facilitate fixation, not using the stabilized images per se, but the pan velocities that come as a result of the process. From the pan velocities of the left and right camera, the expected change in vergence angle, as well as cyclopean gaze direction can easily be found (Pahlavan et al. 1992). For many algorithms, the accuracy deteriorates the further you depart from a certain reference point in the centre of the matching window, which for optical flow could be the same as the point of no motion displacements. When working in an pyramidal framework this is typically the case.

Stabilization may also lead to a more economical execution of the matching operation. The matching range may be such that points close to a particular reference point are given more computational resources, that points located further away. The accuracy should only be high enough for the observer to understand when the reference point ought to be changed. This is indeed the case in many biological systems. For example, when an object is moving in a scene and more information about the object is desired, an observer may change reference point, moving the eyes and head in such a direction that the flow of the object is canceled out. The same thing is true in the case of stereo, where a change in vergence angle, affects the position of zero disparity. It should be emphasized that we do not intend to present a model of how position constancy is achieved in biological systems, but rather use image stabilization to overcome computational limitations of algorithms used. However, it is still interesting to know how it may fit into existing theories on positional constancy.

Non-visual guidance

Our intention is to use visual information in order to stabilize images, but that does not mean that we fully support the idea of direct position constancy as proposed by Gibson (1966). In his theory the structure of the optical flow is used to control stabilization, without the need for additional non-visual information, such as eye movement commands. This is done through a subtraction of the predominant flow component from the optical flow, resulting in a flow field more or less stabilized. However, in biological systems, as well as in most artificial ones, the commands that control head and eyes may be available and could therefore be exploited. Since such commands will be available as soon as the eyes start moving, which is earlier than the resulting optical flow can be observed, it would be advantageous to take them into consideration. Rather, we imagine a system in which stabilization is possible through a combination of the Efferent Copy theory of von Helmholtz (1925), that uses the commands to directly affect stabilization, and methods working on the optical flow.

In our system eye and head commands may be stored and used as initial estimates of the expected future motion displacement. Thus non-visual information could be used to guide methods working on the optical flow, rather than controlling the stabilization exclusively on its own. Using stored commands as initial estimates of displacements, the range of perceived image motions can be extended beyond the high-accuracy range of the methods involved. In order for the commands to be accurately translated into image displacements, some kind of learning has to be performed, at least if the system is to be robust. The residual image displacement after stabilization using to the motor commands, can be used as an error signal to drive such a learning process. How this is to be done in practice is beyond the scope of this thesis. Instead we concentrate on stabilization without using non-visual information, but design the system such that additional information can be taken advantage of, if such is available.

3.1 Related work

Image registration and stabilization has been used for, and is often required by, a wide range of applications, such as ego-motion estimation (Irani et al. 1994*b*), cue integration (Maes et al. 1997), video compression (Stiller & Konrad 1999), image mosaicing (Hansen et al. 1994, Irani et al. 1995, Zoghiami et al. 1997), detection and tracking of moving objects (Davis et al. 1996). There are a number of method that can be used for image stabilization and they differ in the way matching is performed and models used. Comparisons between some different techniques can be found in (Brown 1992, Tian & Huhns 1986), whereas (Morimoto & Chellappa 1997, Balakirsky & Chellappa 1996) study the performance of a number of feature based implementations, in terms of mean square error and moving target detection. Since stabilization has been used for so many different applications, an extensive review of all relevant publications cannot be given in this thesis. A couple of real-time system still ought to be mentioned.

To compensate for motion, Bergen et al. (Bergen, Anandan, Hanna & Hingorani 1992, Hanna 1991) tested a whole series of different motion models; a globally affine model, models based on the assumption of a planar surface or a rigid scene, and a more general model of optical flow. Image data are fitted to the model of choice using the brightness constancy assumption and sums of squared differences. An affine model was also used by Hansen et al. (1994), who implemented a real-time stabilization system using coarse to fine cross-correlations of Laplacians and used the results for mosaic construction. A system based on a 2D rigid motion model has also been presented by Zheng & Chellappa (1993) and implemented in real-time by Morimoto & Chellappa (1996). Features are extracted as peaks in Laplacians of incoming images, with matching performed minimizing sums of squared differences in a coarse to fine framework. The same system has successfully been used for detecting independently moving objects and for camera control (Davis et al. 1996).

Instead of warping images so that two consecutive images overlap, it is in many cases desirable only to compensate for unwanted camera motions that originate from vibrations in the mechanics of the system. The resulting image motion will then reflect the motion of the cameras without such vibrations. Ways of separating these two components can be found in (Yao & Chellappa 1996, Duric & Rosenfeld 1996). It is worth noting that the problems of stabilization and optical flow calculation are indeed related. Both problems are typically solved through a matching operation between two images. Even if models used for optical flow computation are typically applied locally, data may instead be constrained, to fit a more global model of the camera motion (Irani et al. 1994*b*).

3.2 Stabilization methods

In the next part of this chapter three different stabilization methods will be analyzed in terms of accuracy and speed. Stabilization is only done with respect to rotation, since the flow due to translation is typically small in comparison and requires the different depths to be known. The first method is based on corner features and the second one uses image gradients. They are similar to methods that have earlier been presented by others, even if the implementations presented here are different. In a sense the methods represent two extremes on a scale of possible methods, because the first one only uses points that do not suffer from the aperture problem, whereas the last method uses every image point available. In an effort to combine the advantages of both approaches and overcome their respective weaknesses, a third method will be presented. This method uses all points for which image structure exist in at least one dimension. Since such points are more common than corner features, this approach is less likely to collapse, due to lack of local structure.

3.2.1 Stabilization based on corner features

The use of high curvature feature points is natural for stabilizing images, since their exact positions in image space can be determined with relatively high accuracy. Whereas line features only can be aligned from one image to another along the gradient direction, corner features can be aligned in both image dimensions, thus leading to a more constrained problem. Corner features may be found using feature extractors, such as SUSAN (Smith & Brady 1997) or the Harris corner detector (Harris & Stephens 1988). In this section an approach based on such features will be presented, with care taken to make it as fast and robust as possible.

The optical flow due to rotation $\mathbf{u^r}$ can, in terms of the image velocity equation of Longuet-Higgins & Prazdny (1980), be expressed as $\mathbf{u^r} = (u^r, v^r)^\top = \mathbf{B}\omega$, where $\omega = (\omega_x, \omega_y, \omega_z)^\top$ is the rotational speed of the observer and

$$\mathbf{B} = \begin{pmatrix} -xy & 1+x^2 & -y \\ -1-y^2 & xy & x \end{pmatrix} \tag{3.1}$$

depends on the image space position. The rotation ω can readily be solved for using a set of corresponding feature pairs and least squares regression. However, like many similar problems in this thesis, outliers may seriously degrade the performance, unless they have been identified and eliminated from the data set.

In order to quickly and reliably match features between two consecutive image frames, a two-stage process is applied. In the first stage the Mahalanobis distances, in local image intensities and variance, between features from the two images are calculated. As time proceeds statistics are collected from the small (say 7×7 pixel) windows surrounding each corner feature. If the distance is below a threshold the corresponding pixel windows are compared with modified normalized cross-correlation. Assuming that the rotation around the optical axis ω_z is small, the matching scores are then added to a 2-dimensional histogram of ω_x and ω_y. The histogram is composed of 32×32 bins, with a resolution of $2/f$, with f denoting the focal length. Thus the resolution is equivalent to a 2 pixel rotational flow in the origin of the image and the total range is $32/f$, which in our case equals about $4.6°$ per update. After smoothing, the peaks of the histogram are extracted for further processing.

The histogram can be regarded as an approximate log-likelihood diagram over the space of two rotational directions. If information is available either from external non-visual sources or from previous frames, this information may be added to the histogram before peaks are extracted. To extend the search beyond the range defined by the size of the histogram, the centre of the histogram can be redefined using the same information. Previous estimates of ω_z are used such that their contribution to the flow is subtracted from $\mathbf{u_r}$ before histogramming, but the current implementation is limited in the sense that additional non-visual knowledge about ω_z cannot be exploited. Practical experiments have shown that this does not significantly damage the performance of the system.

The histogram peaks taken into consideration are the largest peak together with all other peaks with compiled matching scores more than one half the maximum. Typically this results in between one and three peaks, depending on translational speed and number of independently moving objects in the scene. For each peak, corners from the current to the previous image frames are once again matched, using the position of the peak as the centre of a small search window. In practice no real correlations have to be performed, since the matching scores have been saved from the creation of the histogram. Through this process each feature point will have one corresponding match in the opposite image for each histogram peak.

The same peaks are then used to initialize an M-estimator similar to the one in Section 2.2.3. M-estimators are used to make the process more robust, preventing possible erroneous outliers from dominating the final solution, as would otherwise have been the case if least squares were used. Using a Cauchy function

$$\rho(x,\sigma) = \log(1 + \frac{1}{2}(x/\sigma)^2), \tag{3.2}$$

the following error function is iteratively minimized:

$$E = \sum_i \rho\left(|\mathbf{u}_i^{\mathbf{r}} - \mathbf{B}_i\omega|,\ \sigma\right). \tag{3.3}$$

Rewriting E as the a corresponding iterated reweighted least-squares problem

$$E_k = \sum_i w_{i,k}(\mathbf{u}_i^{\mathbf{r}} - \mathbf{B}_i\omega^k)^2, \tag{3.4}$$

the minimization is performed using weights defined by the errors of the previous iteration, $\epsilon_{i,k} = |\mathbf{u}_i^{\mathbf{r}} - \mathbf{B}_i\omega^{k-1}|$ and the weight function

$$w_{i,k} = \frac{1}{\sigma^2 + \epsilon_{i,k}^2/2}. \tag{3.5}$$

The shape parameter σ^2 is updated such that the number of feature pairs with squared errors below $8\sigma^2$, $N_{e^2<8\sigma^2}$, gradually reaches about λN_{tot}, where N_{tot} is the total number of pairs and λ is a fraction between 0 and 1. In the current implementation λ is set to 0.5, assuming that at least 50% of the feature pairs have been correctly matched and do not belong to a independently moving foreground object. The update function of σ^2 is

$$\sigma_{k+1}^2 = \lambda \frac{N_{tot}}{N_{e^2<8\sigma^2}+1}\ \sigma_k^2. \tag{3.6}$$

After this iterative process has been completed for each initial histogram peak, a solution corresponding to the one with the smallest σ^2 is chosen as the

```
Stabilization using corner features

1. Find corner match candidates using Mahalanobis distance.
2. If the distance small enough,
3.     Match corners using modified normalized cross-correlation.
4. Store matches in histogram.
5. For each histogram peak,
6.     Match corners using modified normalized cross-correlation.
7.     Find rotation using M-estimators.
8. Choose result corresponding to the lowest error.
```

Figure 3.1. Stabilization using corner features

final result. The algorithm may be summarized with the pseudo-code given in Figure 3.1. For a 384×288 image, the total computational cost in the case of one peak on a 1.2 GHz Athlon MP processor is approximately 2.8 ms and an additional 0.8 ms required for each additional peak. The cost of extracting corner features, which is about 3.0 ms following the suggestions in the appendix, has not been included, since corners are expected to be required by other parts of the complete system.

Performing stabilization using corner features can be done reliably, as long as enough such features are available in the scene, which is not necessarily the case. Scenes like the one in Figure 3.2 are common in indoor environments. It could also be that features exist due to occlusions, which means that they do not represent real corners in 3D space. They might also appear and disappear behind objects in the foreground, complicating the matching problem. In an attempt to overcome this weakness, another method based on image gradients will be presented in the next section.

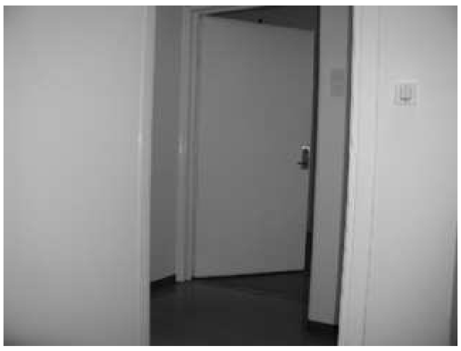

Figure 3.2. A hallway as seen by an autonomous observer

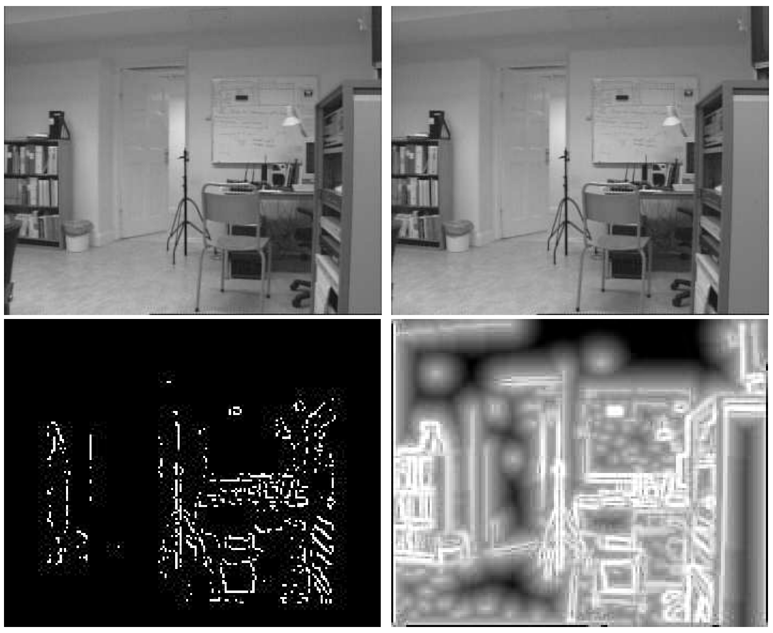

Figure 3.3. An example of two consecutive images (top), with extracted contour points of the first image (left) and the gradient magnitude of the second image (right), with regions between contours filled-in with negative magnitude values.

3.2.2 Stabilization based on contour points

Instead of stabilizing images using high curvature feature points, the method presented in this section uses all image points with high enough gradients, even if structure only exists in one dimension, which is true for line features. Similar to Canny's edge detector (Canny 1986), edge pixels are found localizing points of maximum gradient magnitude. First the magnitude has to be large enough and then it has to reach a local maximum in the gradient direction. Simply thresholding the magnitudes typically results in broken edge contours. Canny solves this problem through a hysteresis operation using another lower threshold, filling in the gaps of the broken contours. Since the method presented here does not rely on the extraction of complete contours, this phase is simply ignored. A result of that can be seen in the left column of Figure 3.3, where the threshold on the gradient magnitude τ_1, has been set high enough, such that only a sufficient number of points are found for the stablization algorithm to be successful.

The idea is to match these contour points from the first image (lower left of Figure 3.3) to the gradient magnitudes in the second image (lower right) and maximize the total sum of magnitudes, which will be regarded as the matching strength between the two images. Since the summation is performed only over contour points, the matching is considerably faster than if is was done over every

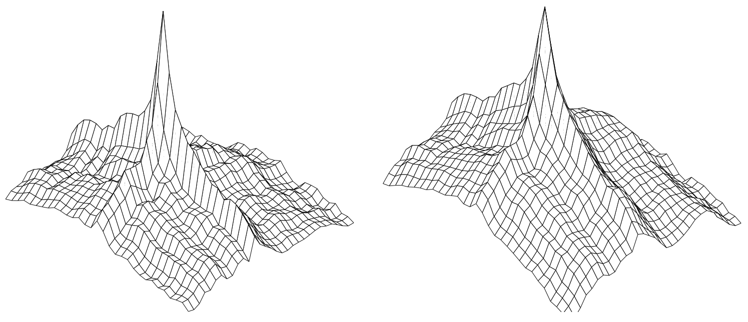

Figure 3.4. Total matching strength as a function of ω_x and ω_y, with (right) and without (left) filling structure-less regions with negative magnitude values.

pixel. A graph showing the matching strengths for different combinations of ω_x and ω_y can be seen in the left image of Figure 3.4, which shows the strengths for angles within 4.3° from the peak. Even if the peak of the correct rotation is distinct, the surface unfortunately includes a number of additional maxima. Outside the peak the surface is more or less flat, which makes gradient ascent in the space of rotations hard to achieve. In order to guide a search to the correct maximum, image regions between contours are instead filled-in with negative magnitude values, so as to make the matching surface less flat, as shown in the right image of Figure 3.4.

Computationally this is done by first thresholding the gradient magnitude of the second image and gradually letting negative magnitudes grow out from the contour points, defined by magnitudes greater than τ_2. An example of that can be seen in the lower right image of Figure 3.3. This threshold is about one third of the one used for the contour points in the first image, that is τ_1. Each pixel that does not belong to a contour keeps track of its closest contour point and is assigned a negative magnitude value linearly dependent on the distance to this point. Only points within a predefined maximum distance are considered. Even if a quadratic dependence would be preferable, since that would typically lead to a fast convergence, a linearly increasing cost is more robust.

In order to minimize the computational cost, the filling operation is performed such that each pixel is only visited once. A list of active image points, that have recently been assigned a magnitude value, are kept and read in the order of closest distance registered. Initially the list only includes the extracted contour points. Even if the information stored from the two images is different in its nature, the collection of data is performed simultaneously, so that in the next time frame contour points have already been extracted for the first image.

In order to locate the peak of maximum matching strength a cloud of initial starting points is spread around a centre defined by the previously estimated rotation or an a priori assumption based on non-visual information. For each starting point a local maximum is then sought through an iterative ascent in directions of ω_x, ω_y and ω_z, one rotational axis after the other. Steps are taken such that the resulting flow is equivalent to an average of about one pixel. For each starting point the iterations are continued until either a local maximum has been found or a point outside the range of reasonable rotations is reached.

Once the local maxima have been found, the rotational estimates are improved using image gradients, instead of the gradient magnitudes. The reason for doing so is that the method presented above tends to be biased towards rotational flows of integer displacement. With $I_{x,i}$, $I_{y,i}$ and $I_{t,i}$ denoting the gradients of an image point (x_i, y_i) and $\mathbf{B}_i$ the corresponding flow directions according to Equation 3.1, a minimization is performed on the squared error

$$E = \sum_i (\, [I_{x,i}, I_{y,i}] \, \mathbf{B}_i \omega + I_{t,i} \,)^2. \tag{3.7}$$

The summation is done only on the high gradient contour points, for the sake of maximum speed. In order to make the approach robust and dismiss contours originating from independent motion, only points where high gradients exist in both images, are taken into consideration. The total computational cost is approximately 7 ms on images of size 192 × 144 pixels, but may vary depending on the number of initial rotations used and local maxima found. The complete algorithm can be summarized with the pseudo-code in Figure 3.5.

Stabilization using contour points

```
1. Find contour points in first image with gradient magnitude > $\tau_1$ and
   maximum in the gradient direction
2. Find contour points in second image with gradient magnitude > $\tau_2$
3. Fill regions between contours in second image with negative gradients
4. For each randomly generated initial rotation
5.     Until a local maximum is found or outside range
6.         Ascend sequentially in the directions of $\omega_x$, $\omega_y$ and $\omega_z$
7. For all local maxima large enough
8.     Estimate rotation using least squares on image gradients
9. Pick the result with the lowest residual error
```

Figure 3.5. Stabilization using contour points

3.2.3 Stabilization using image gradients

The third method implemented is similar to that of Hanna (1991) in that the brightness constancy assumption is used in conjunction with a model of flow induced by 3D camera motion. However, instead of including flow due to translation, representing the scene as a planar patch, only the rotational component is considered, similar to the methods described in previous sections. Through practical experiments of an autonomous platform in an indoor environment it has been observed that the translational component is typically very small. We have instead chosen to compensate for rotations only and consider a resulting residual as being part of either the translational flow or due to objects of independent motion. For other applications, such as high-speed car control (Dickmanns 1997), the situation might be very different.

The rotational component is found minimizing the same function as in the last step of the previously presented method, that is Equation 3.7. Operations are performed in a three level coarse to fine framework, where the finest scale in the current implementation is 192 $\times$ 144 pixels in size. Three iterations are used for each level, except for the finest scale that only includes one. Results are warped between levels as well as iterations. Through the warping procedure, the temporal image derivative I_t of iteration $k+1$ is estimated as

$$I_t = I_2(x + u_k, y + v_k) - I_1(x, y) - [I_x, I_y]\mathbf{u_k}, \tag{3.8}$$

where $\mathbf{u_k} = (u_k, v_k)^\top = \mathbf{B}\omega_k$ is the flow due to the previously estimated rotation and, I_1 and I_2 are the two consecutive images.

From the coarse to fine process, the maximum possible range of rotations, measured in the centre of the image, can be expected to be about $2^2{*}3{+}2^1{*}3{+}1 = 19$ pixels at the finest scale, which is equivalent to approximately 5.4° per update. However, in order for such a large rotation to be determined, the largest possible rotation must be found at every scale. This is not likely in practice, at least not if rotations are to be estimated reliably. Since errors are most likely near the limits of what the algorithm allows, and because errors from coarser levels are magnified for each new finer scale, estimates of large rotations can be expected to be erroneous. In order to correct for this weakness multiple initial values of ω_x and ω_y are instead used at the coarsest scale. An estimate corresponding to the lowest residual error is then passed to the next iterations. The complete procedure has a computational cost of about 9.6 ms. The cost is higher than that of the two other methods, even if this method was not implemented robustly.

3.3 Experiments

In order to test the presented methods, a series of 126 different indoor images were considered. Each such image was rotated using a known rotation, creating 126 pairs of images. For each such pair, the corresponding rotation angle was estimated using the three stabilization methods. This operation was performed for different true rotation angles, between 0° and 4.6° around an axis in the image plane and for each angle statistics were collected. For each method, the standard deviation as a function of the rotation angle can be seen in Figure 3.6. Since the last step of the method using contour points, is similar to the one using image gradients, the results are quite similar. There is, however, a difference in bias between the two methods. Whereas the former method includes no noticeable bias, a linearly increasing bias, with a maximum of 0.06°, can be observed for

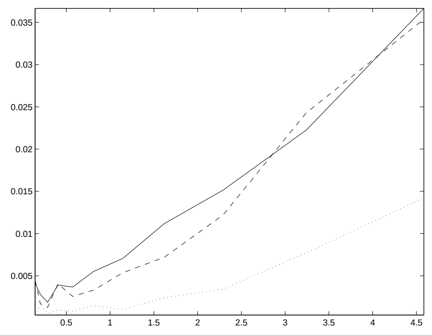

Figure 3.6. The standard deviation of estimated rotations in degrees as a function of rotation magnitude, for the gradient-based method (solid), the one using contour points (dashed) and corner features (dotted).

the gradient based method. An explanation for this bias is that spatial image derivatives in this approach are calculated using the first image only, while the contour based method uses the average between the two images. However, bias as well as standard deviation are relatively small for all three methods, at least for our application.

It should be noted that these results are the best one should expect. In a real situation, with two images taken by the same cameras from two different instances in time, additional errors are likely to be present from a number of reasons. First of all, illumination might differ, globally as well as locally. Such errors will probably be fewer for the feature based methods. With rapidly rotating cameras, motion blur is evident in typical images. In the ground-truth data there are no independently moving objects and no camera translations that might also have degraded the performance.

Rotation $\lvert\omega\rvert$	Corner (3.2.1)	Contour (3.2.2)	Gradient (3.2.3)
0.9°	2.5%	2.6%	2.7%
1.8°	2.6%	2.9%	3.2%
2.4°	2.6%	3.0%	3.7%
3.6°	2.6%	3.3%	3.9%

Figure 3.7. The fractions of pixels with absolute errors in pixel values exceeding 8, after warping and subtraction, for different stabilization methods.

Experiments on real sequences

Evaluations were also performed using a series of real images, letting a binocular head pan from right to left at a constant speed. For each image, the fraction of image points with pixel error after warping of more than 8 was calculated, assuming the maximum pixel value to be 255. The threshold was set so as to

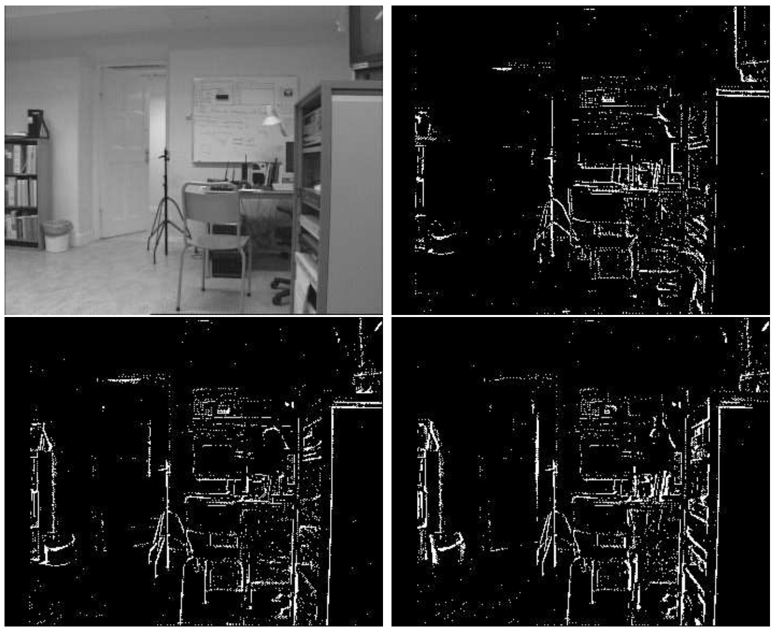

Figure 3.8. An image extracted while letting a camera pan from right to left (upper-left) and images showing points with errors in pixel value more than 8 after stabilization, for methods using corner features (upper-right), contour points (lower-left) and image gradients (lower-right).

discard typical image noise, but still accept errors due to failures in the stabilization. Some errors can also be expected in the warping procedure, especially near high gradient edges. Results are shown in Figure 3.7. As can be seen the corner based method is relatively insensitive to the magnitude of rotation, whereas especially the gradient method suffers significantly, but for no method does the stabilization fail completely.

For rotations of more than 4.5° all methods start to fail for an increasing number of test cases. The gradient based method is the last method to collapse for large rotations, while the corner based one has problems as soon as rotations are beyond the range of the histogram used (see Section 3.2.1). The approach using contour points rarely performs as well as the one based on corners, but always better than the gradient based one. An example showing points of pixel errors after stabilization can be seen in Figure 3.8. The example images are chosen such that they reflect the average performance of the methods. As can be seen most errors do not belong to large regions of erroneous pixels. The low error limit is sometimes exceeded by quantization noise, mainly at object boundaries. However, the errors due to the stabilization process are few.

The next sequence, out of which three images are shown in the first row of Figure 3.9, contains a walking person in front of a moving camera, located on

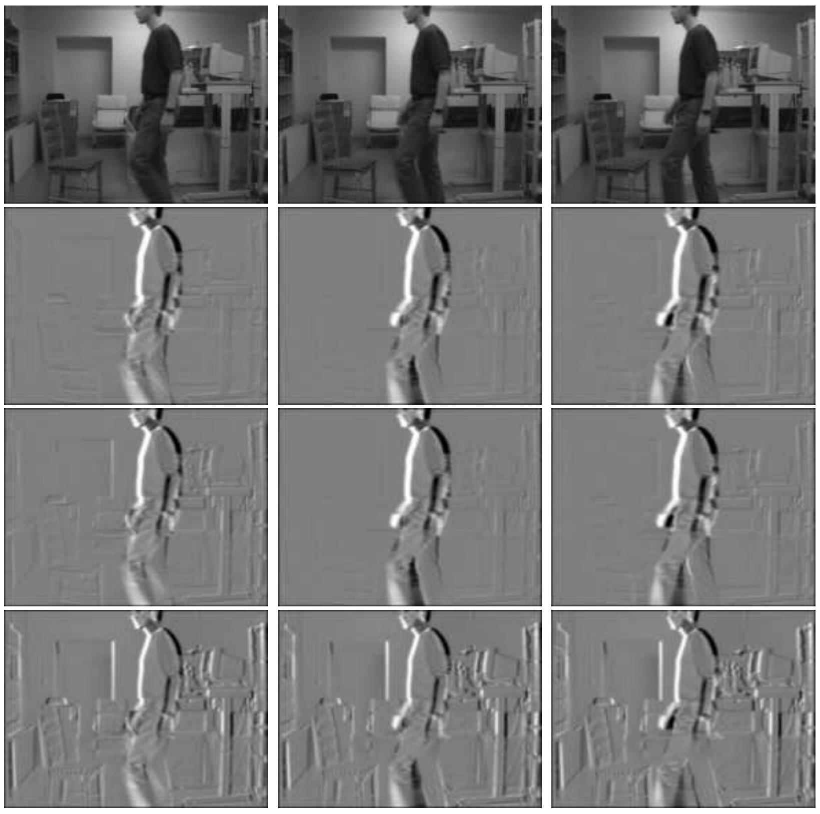
Figure 3.9. Images taken from a camera translating forward and residuals after stabilization for the methods based on corners features (2nd row), contour points (3rd row) and image gradients (last row).

top of an autonomous platform translating forward at a speed of approximately 0.1 m/s, without any rotations. This is a more challenging sequence than the previous one, since image motion due to translations of the person and the platform might interfere with the stabilization process. Results from the three methods can be seen on the remaining rows of the same figure. It is clear that all methods, except the method based on image gradients, does a rather good job at identifying the background motion.

Since the estimated rotation should be zero for all tested images, the rotational errors can be estimated. The errors are tabulated in Figure 3.10, together with fractions of points exceeding 8 in pixel values. From the results of the gradient based method one may conclude that more errors are related to the bias. When analyzed in detail, this method proves to be sensitive to independent motion in the scene. While the other methods are designed such that irrelevant corners and contours of an independently moving object in the foreground are

	Corner (3.2.1)	Contour (3.2.2)	Gradient (3.2.3)
Erroneous pixels	8.6%	9.8%	15.0%
Rotation bias	0.01°	0.03°	0.67°
Rotation error	0.05°	0.09°	0.73°

Figure 3.10. The fraction of points with errors larger than 8 in pixel values, rotational errors and bias for the tested methods.

disregarded, the gradient based method tries to find a camera rotation that satisfies foreground as well as background, resulting in a motion that matches neither. Better results can be expected if the method is made robust, but that would also make it slower and it is already the slowest one among the three tested methods.

The final example uses the sequence from Section 2.3 with optical flow shown in Figure 2.7. The camera undergoes a 1.4° rotation per update or 35°/s, while translating forwards. The motion is such that the walking person is tracked in the centre of the camera. The residuals after stabilization can be seen in Figure 3.11. Even if the translation is as large as about 0.85 m/s and the platform in the foreground is located only 1.8 metres away, the performance of the stabilization is similar to that of the previous case. The major difference is an increased variance of the rotational estimates due to the gradient based method.

The images in Figure 3.12 illustrates the performance of Lucas & Kanade's optical flow method when the background has been stabilized using the corner based method. The upper row shows the results of the sequence just mentioned. Since the optical flow is well within the bounds of what the optical flow calculation can handle, there is not much difference, even if a slight improvement in the uniformity of the background can be noticed. The second row shows results from the panning sequence in Figure 3.8 with the camera rotating at a speed of 3.6° per update. In this case the optical flow is too large for Lucas & Kanade's method. For this particular sequence, an increased number of levels in the pyramid could be a solution, but that would result in an undesirable blurring under other circumstances.

3.4 Conclusions

Stabilization is a beneficial, and sometimes even necessary, component in a range of applications, such as cue integration and tracking of moving objects. The knowledge of dominanant image motion can be exploited in a number of ways. In this chapter three different methods for compensating camera rotation have been explored; one based on corner features, another using contour points without requiring contours to be explicitly available, and finally a commonly used method based on image gradients. The first two methods are novel in the way they combined robustness with speed. All three methods perform well in cases of limited camera translations and no independent motion. In more difficult situations, the corner and contour based methods are still able to separate these

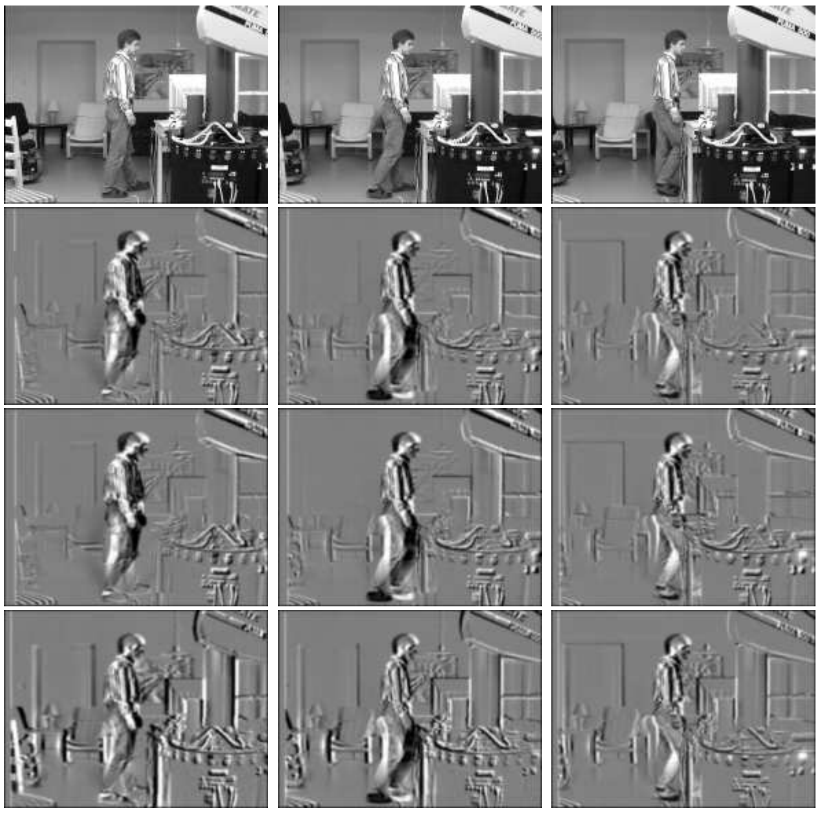

Figure 3.11. Images taken from a rotating and translating camera (upper row) and residuals after stabilization for the methods based on corner features (2nd row), contour points (3rd row) and image gradients (last row).

components from camera rotations, but the gradient based method, due to a lack of a robust formulation, is not.

It would be worth investigating whether the gradient based method can be made robust, without introducing a significant increase in complexity and computational cost. The system designed during the work of this thesis uses the methods based on corner as well as contour points. Corners are first extracted and depending on the distribution and number of corners found, the appropriate stabilization method is chosen. Since corners are used for both stereo and motion analysis, stabilization using corners results in considerably lower computational cost. However, there are more opportunities for possible speed-ups in the case of the contour point method. The sum of gradient magnitudes in the optimization of the contour based method gives a good indication of what maximum matching strength to expect, and if a good initial guess is available it is possible to directly locate the global maximum.

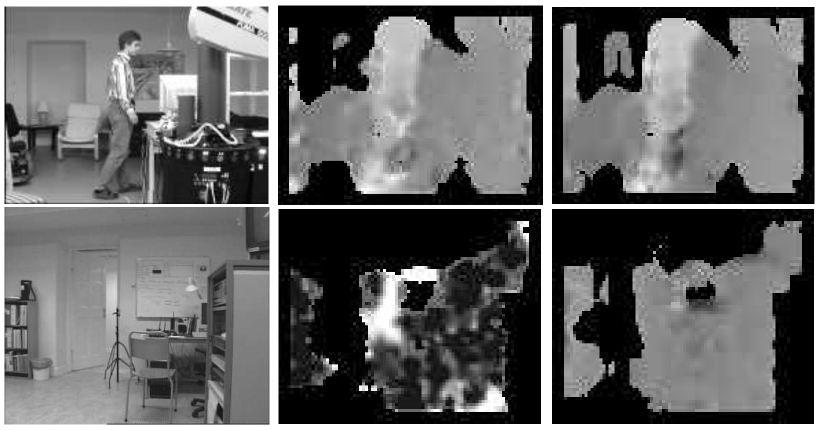

Figure 3.12. Optical flow calculated for sequences in Figures 3.11 and 3.8, with (right column) and without (middle column) stabilization of background.

Chapter 4

Epipolar geometry

An active observer would benefit from the use of binocular stereo. The main reason is that disparity is such a strong cue to depth. However, in order to relate disparities to distances in 3D space the relative positions and orientations of the cameras, the epipolar geometry, has to be determined. This chapter deals with the problem of estimating the epipolar geometry, as fast and robustly as possible. After an introductory motivation, the so called epipolar constraint will be explained in terms of a matrix known as the essential matrix. A number of additional constraints will be introduced in order to minimize the complexity of the problem. A process for identifying and eliminating of outliers is proposed and experiments are performed in order to evaluate the different models. An iterative method based on the bilinear optical flow equation will also be presented. It will be shown that this approach leads to more robust results, even if it is based on an approximate model. Finally, a couple of real-time experiments will be performed and the performance analyzed.

4.1 Dynamic vergence

An observer working in a dynamic world is typically involved in tasks that require interaction with objects at different depths. In the case of manipulation these objects will be located close to the observer, whereas navigation around the scene leads to interactions with objects further away. If the visual field is relatively small, which is most often the case, the observer will be forced to dynamically verge its cameras, so that objects of interest are visible by both cameras. Computationally, image locations close to the border of the visual field often suffer from filter truncation errors, especially in multi-scale frameworks. This means that objects of interest should preferably be kept at the centre of the images, where such errors do not exist. Furthermore, in foveated systems most computational resources are concentrated to the centre of the visual field, which is another reason for objects to be centered.

An additional reason why a stereo system would benefit from dynamic vergence is the fact that fixation might simplify computational tasks such as calculating the ego-motion and the motion of an object present in the scene (Fermüller & Aloimonos 1993, Ballard 1991, Daniilidis & Thomas 1996). Fixation implies that the camera system is not just verged, but that the optical axes also intersect somewhere in front of the cameras. Hence, there is no relative tilt between cameras. If the highest possible accuracy is desired when calculating motion and shape, the object of interest should be located close to this intersection of the optical axes, the fixation point.

If the cameras are verged and depths are to be determined, it is essential that the relative positions and orientations of the cameras are known. If this is not the case, it is not possible to relate calculated disparities to actual distances in 3D space. One possibility is using counters on the motors controlling the motion of the stereo head system. However, there are several reasons why this is not always feasible. First of all, if the vergence angle, that is the difference in orientation between the cameras keeps changing, it is necessary to synchronize image capture with the readings from the motor counters. In practice, this is not easy at all. The system controlling the camera motions and the one capturing images might be physically separated, communicating only through a link with a latency that is hard to predict. The system would most likely be simpler as well as cheaper if the camera configuration could be estimated using image information only.

Even if motor positions are available at every instance in time, it is still questionable if the information can be fully relied upon. As the observer is moving around in the environment, it is hard to prevent the cameras from vibrating. A system able to dynamically verge typically consists of a series of moving parts, which means that it is very sensitive to disturbances. Small vibrations easily corrupt the calculated disparities, leading to large errors in estimated depth. It would be possible to add additional weight to the stereo head system making it more robust and less sensitive to vibrations, but such a system would probably be more difficult to control and slower. In any case, motor information from body motions, in our case when the entire robot moves, will be too inexact to use to control camera movements.

A last reason why the camera configuration ought to be estimated using image information, is that the results may be used for further operations on the same images. For example, in the system presented here information about the camera configuration will be used in order to perform a rectification of the images, that is rotating the images so that the rotated images will look as if the cameras were located in parallel. If the estimated configuration is based on image data, it is easier to relate the errors to image noise, than if only motor counter information were used. However, this does not mean that external information cannot be exploited at all. Estimation of the camera configuration using image data may be easier and faster if an approximate configuration is already known. Such information might be available from monocular stabilization of the left and right images, as described in Chapter 3.

In the system presented here the camera configuration, represented by the epipolar geometry in later sections, will be estimated continuously. Internal camera parameters such as focal length, optical centers and scaling are supposed to be already calibrated, and recalibrated only when changed. These parameters are not as sensitive to external disturbances and thus assumed to be constant in the remainder of this chapter.

4.2 The essential matrix

Consider a binocular system consisting of two cameras with centres positioned at $\mathbf{c_l}$ and $\mathbf{c_r}$, separated by a baseline $\mathbf{t} = \mathbf{c_r} - \mathbf{c_l}$. The camera centres together with a point in 3D space $\mathbf{u}$ define a plane, as shown in Figure 4.1. The two vector $\hat{\mathbf{x}}_\mathbf{l} = \mathbf{u} - \mathbf{c_l}$ and $\hat{\mathbf{x}}_\mathbf{r} = \mathbf{u} - \mathbf{c_r}$, representing the projections of this point onto the left and right cameras, will lie on this plane, as will the baseline $\mathbf{t}$. This means that the determinant $D = [\hat{\mathbf{x}}_\mathbf{l}, \mathbf{t}, \hat{\mathbf{x}}_\mathbf{r}] = 0$, which is known as the epipolar constraint and was introduced independently by Longuet-Higgins (1981) and Tsai & Huang (1981). The two projections can be given in the local frames of each camera using the rotation matrices, $\mathbf{R_l}$ and $\mathbf{R_r}$, that describe the orientation of the cameras relative to a reference frame. Thus $\mathbf{x_l} = \mathbf{R_l}\hat{\mathbf{x}}_\mathbf{l}$ and $\mathbf{x_r} = \mathbf{R_r}\hat{\mathbf{x}}_\mathbf{r}$ denote the vectors in this reference frame.

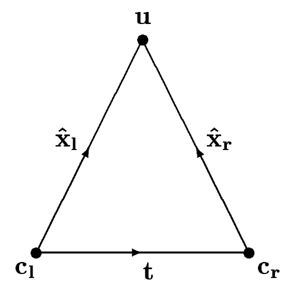

Figure 4.1. An epipolar plane

With $\mathbf{T}$ being the skew-symmetric matrix representing an outer product of $\mathbf{t}$ and some other vector $\mathbf{a}$, that is $\mathbf{Ta} = \mathbf{t} \wedge \mathbf{a}$, the determinant may be written as

$$D = [\hat{\mathbf{x}}_\mathbf{l}, \mathbf{t}, \hat{\mathbf{x}}_\mathbf{r}] = |\hat{\mathbf{x}}_\mathbf{l} \cdot \mathbf{t} \wedge \hat{\mathbf{x}}_\mathbf{r}| = \hat{\mathbf{x}}_\mathbf{l}^\top \mathbf{T} \hat{\mathbf{x}}_\mathbf{r} = \mathbf{x}_\mathbf{l}^\top \mathbf{R_l} \mathbf{T} \mathbf{R}_\mathbf{r}^\top \mathbf{x_r} = \mathbf{x}_\mathbf{l}^\top \mathbf{E} \mathbf{x_r} = 0. \quad (4.1)$$

The 3×3 matrix $\mathbf{E} = \mathbf{R_l} \mathbf{T} \mathbf{R}_\mathbf{r}^\top$ is often called the essential matrix and has a number of interesting properties. The product $\mathbf{l_r} = \mathbf{E}\mathbf{x_r}$ represents a line in the left image and since $\mathbf{x}_\mathbf{l}^\top \mathbf{l_r} = 0$ the image point $\mathbf{x_l}$ lies somewhere along this line. Thus a point in the right image will constrain the corresponding point in the left image to a line, given that the epipolar geometry is known. The converse is also true, in that $\mathbf{x_r}$ will be constrained to another line $\mathbf{l_l} = \mathbf{E}^\top \mathbf{x_l}$ in the right image. The major benefit of this constraint is that it relates image features to each other, without knowing the actual 3D point being projected. The problem of finding positions in 3D space has efficiently been decoupled from the problem of estimating the camera configuration.

The essential matrix can be factored in two different components, $\mathbf{E} = \mathbf{R}\mathbf{S}$, where $\mathbf{R} = \mathbf{R_l}\mathbf{R_r}^\top$ is a rotation matrix and $\mathbf{S} = \mathbf{R_r}\mathbf{T}\mathbf{R_r}^\top$ is skew-symmetric. The second component $\mathbf{S}$ is just $\mathbf{T}$ transformed into the right camera frame, that is an outer product of $\mathbf{s} = \mathbf{R_r}\mathbf{t}$ and some vector. The product $\mathbf{s} \wedge \mathbf{x_r}$ is the plane given by the right image point and the baseline, and $\mathbf{R}$ transforms this plane into the reference frame of the left camera. Given that the epipolar geometry constraint is satisfied, that is $\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r} = 0$, the point $\mathbf{x_l}$ will lie on the same plane.

The essential matrix is of rank-2, which is easily seen analyzing the singular values of $\mathbf{E}$. The square of $\mathbf{E}$ can be written as follows:

$$\mathbf{E^2} = \mathbf{E}^\top\mathbf{E} = \mathbf{S}^\top\mathbf{R}^\top\mathbf{R}\mathbf{S} = \mathbf{S}^\top\mathbf{S} = (\mathbf{s}^\top\mathbf{s})\mathbf{I} - \mathbf{s}\mathbf{s}^\top. \tag{4.2}$$

Since this matrix annihilates $\mathbf{s}$, one of the singular values must be 0. However, any vector $\mathbf{u}$ orthogonal to $\mathbf{s}$ is an eigenvector of $\mathbf{E^2}$, since $\mathbf{E^2}\mathbf{u} = (\mathbf{s}^\top\mathbf{s})\mathbf{u}$. Thus $\mathbf{E}$ has two singular values equal to $\mathbf{s}^\top\mathbf{s}$ and the essential matrix is of rank-2. In fact, it can be shown (Tsai & Huang 1984, Huang & Faugeras 1989) that a matrix can be factored into a rotation and a skew-symmetric matrix if and only if it has two equal non-zero singular values and one equal to 0. This fact will later be used when the epipolar geometry of the stereo head system is determined.

4.2.1 The 8-point method

Suppose that a number of image feature pairs are given and one would like to estimate the relative orientation and position of the two cameras. In order to do this, one could search for a matrix $\mathbf{E}$ that satisfies the epipolar constraint $\mathbf{x_l}^\top\mathbf{E}\mathbf{x_r} = 0$, where $\mathbf{x_l} = (x_l, y_l, f_l)^\top$ and $\mathbf{x_r} = (x_r, y_r, f_r)^\top$ are the left and right projections of a given 3D point. Since the matrix multiplied by a factor k will satisfy the same constraint, the translation $\mathbf{t}$ can only be determined up to scale. This means the total number of unknown degrees of freedom is five, three rotational and two translational ones.

Due to noise and outliers in the data set it will never be possible to find a matrix that perfectly satisfies the epipolar constraint for each existing feature pair, at least if there are more than five. Instead one could determine an estimate

$$\mathbf{E_9} = \begin{pmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{pmatrix} \tag{4.3}$$

of the true essential matrix $\mathbf{E}$, that minimizes the algebraic error $|\mathbf{x_l}^\top\mathbf{E}\mathbf{x_r}|$. How this can done in practice will be covered later on. But once such an estimate has been found, how can we make sure that it indeed is a essential matrix? After all, $\mathbf{E_9}$ is a 3×3 matrix and consists of nine different elements. Disregarding the factor of scale, it has got eight degrees of freedom and not five.

Based on a scheme suggested by Toscani & Faugeras (1986), a projection can be performed from the original space of eight dimensions, to the five dimensional manifold spanned by all possible essential matrices. Let $\mathbf{E_9}$ be factored into its singular value decomposition (SVD), that is $\mathbf{E_9} = \mathbf{UDV}^\top$. The diagonal matrix $\mathbf{D} = \text{diag}(r, s, t)$ consists of the three singular values. As mentioned above an essential matrix should have two equal non-zero singular values and a last value equal to zero. An essential matrix can then be found, simply by substituting $\mathbf{D}$ with $\hat{\mathbf{D}} = \text{diag}(k, k, 0)$. If $k = (r + s)/2$, the resulting matrix $\hat{\mathbf{E}} = \mathbf{U}\hat{\mathbf{D}}\mathbf{V}^\top$ will be the essential matrix closest to $\mathbf{E_9}$ under the Frobenius norm, that can be decomposed into $\hat{\mathbf{E}} = \mathbf{RS}$, which is a necessary condition for $\hat{\mathbf{E}}$ to be an essential matrix as shown above.

The factorization of $\hat{\mathbf{E}}$ into $\mathbf{RS}$ can be directly found from the singular value decomposition. With the matrices

$$\mathbf{Y} = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{Z} = \begin{pmatrix} 0 & -k & 0 \\ k & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \tag{4.4}$$

the rotational factor can be shown to be either $\mathbf{R} = \mathbf{UYV}^\top$ or $\mathbf{R} = \mathbf{UY}^\top\mathbf{V}^\top$. The difference between the two alternatives is a 180° rotation around the baseline. The configuration for which all triangulated 3D points are located in front of the cameras should preferably be chosen, since no point may exist behind the camera centres. There are also two solutions for the translation, $\mathbf{S} = \mathbf{VZV}^\top$ or $\mathbf{S} = \mathbf{VZ}^\top\mathbf{V}^\top$, which may be separated using a similar consideration.

Counting the degrees of freedom of $\hat{\mathbf{E}}$ leads to a total of seven, three from each orthogonal matrix $\mathbf{U}$ and $\mathbf{V}$, and one from $\hat{\mathbf{D}}$. However, since the essential matrix can only be estimated up to scale, k may be set to 1. Also since the two equal non-zero singular values of $\hat{\mathbf{E}}$ are equal, the estimate is insensitive to a rotation of $\mathbf{V}$ around $\mathbf{v_3}$, that is the last column of $\mathbf{V}$. Thus the total number of degrees is not seven, but five.

4.3 A typical stereo head

Finding the essential matrix as described above might not be the best approach if the camera configuration is known to be constrained. For example, stereo head systems, where two cameras are mounted on each side of a fixed baseline, are often used so that they always are at fixation, that is the optical axes of the cameras intersect somewhere in front of the cameras. Thus there is no relative tilt between the cameras. Furthermore, it is usually not necessary to rotate the cameras around their optical axes. After all, such a rotation would not change the image data, only its orientation. A joint tilt of both cameras does not change the nature of the problem, since the epipolar constraint describes the relative orientation between the cameras, not between the cameras and some reference frame. This means that in reality we may only have two degrees of freedom, one pan rotation for each camera. However, even if the stereo head is constrained, the flexibility of the system would not be affected. The described system can be seen as one of the special cases in a review of configurations by Brooks et al. (1996).

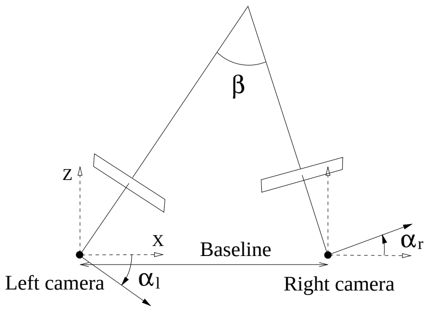

Figure 4.2. stereo head system

In order to describe the actual geometry, a coordinate system is needed. The baseline between the two cameras may be used to define a x-axis, as seen in Figure 4.2. As mentioned earlier a stereo head in fixation has the two optical axes intersecting somewhere in front of the cameras. This point together with the two camera centres define a plane and one may the use the normal of this plane as the y-axis, while the z-axis is located on the plane and perpendicular to the baseline. In this coordinate system the rotations of the cameras can be described by the following matrices

$$\mathbf{R_l} = \begin{pmatrix} \cos(\alpha_l) & 0 & \sin(\alpha_l) \\ 0 & 1 & 0 \\ -\sin(\alpha_l) & 0 & \cos(\alpha_l) \end{pmatrix} \tag{4.5}$$

and

$$\mathbf{R_r} = \begin{pmatrix} \cos(\alpha_r) & 0 & -\sin(\alpha_r) \\ 0 & 1 & 0 \\ \sin(\alpha_r) & 0 & \cos(\alpha_r) \end{pmatrix}, \tag{4.6}$$

where α_l and α_r are the pan angles of each camera respectively and $\beta = \alpha_l + \alpha_r$ is the vergence angle between the two cameras. The baseline is given by $\mathbf{t} = (k, 0, 0)^\top$ and its corresponding skew-symmetric matrix

$$\mathbf{T} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & k \\ 0 & -k & 0 \end{pmatrix}, \tag{4.7}$$

which represents an outer product of $\mathbf{t}$ and some other vector, as was mentioned earlier. Following the derivation in Section 4.2, the essential matrix $\mathbf{E} = \mathbf{R_l}\mathbf{T}\mathbf{R_r}^\top$ can now be expressed as

$$\mathbf{E} = k \begin{pmatrix} 0 & -\sin(\alpha_l) & 0 \\ -\sin(\alpha_r) & 0 & \cos(\alpha_r) \\ 0 & -\cos(\alpha_l) & 0 \end{pmatrix}. \tag{4.8}$$

Since the epipolar geometry only can be determined up to scale, the factor k may be set to 1. Thus the essential matrix lies in a two-dimensional space of possible epipolar constraints, even if the matrix itself involves four unknown elements.

4.3.1 The constrained case

In order to estimate the true essential matrix $\mathbf{E}$, the two angles α_l and α_r have to be determined. This could be done solving a non-linear optimization problem. However, such a problem would typically rely on a number of iterations and an initial guess close to the true solution. An alternative linear approach is finding the matrix $\mathbf{E_4}$ that minimizes the error $\mid \mathbf{x_l}^\top \mathbf{E} \mathbf{x_r} \mid$, where

$$\mathbf{E_4} = \begin{pmatrix} 0 & c_1 & 0 \\ c_2 & 0 & c_4 \\ 0 & c_3 & 0 \end{pmatrix}. \tag{4.9}$$

After finding an optimum in the four-dimensional space spanned by the parameters c_j, $j = 1 \ldots 4$, a projection onto the two-dimensional manifold of possible epipolar constraints is done, similar to the unconstrained case covered earlier. Using the approach suggested by Toscani & Faugeras (1986), the singular value decomposition of $\mathbf{E_4}$ can be shown to be $\mathbf{UDV}^\top$, where

$$\mathbf{U} = \frac{1}{\sigma_1} \begin{pmatrix} c_1 & 0 & c_3 \\ 0 & \sigma_1 & 0 \\ c_3 & 0 & -c_1 \end{pmatrix}, \qquad \mathbf{V} = \frac{1}{\sigma_2} \begin{pmatrix} 0 & c_2 & -c_4 \\ \sigma_2 & 0 & 0 \\ 0 & c_4 & c_2 \end{pmatrix} \tag{4.10}$$

and $\mathbf{D}$ is a diagonal matrix with singular values equal to $\sigma_1 = \sqrt{c_1^2 + c_3^2}$, $\sigma_2 = \sqrt{c_2^2 + c_4^2}$ and 0. An essential matrix is found, substituting the epipolar geometry $\mathbf{E_4}$ with an approximation $\hat{\mathbf{E}} = \mathbf{U}\hat{\mathbf{D}}\mathbf{V}^\top$, where $\hat{\mathbf{D}} = \mathrm{diag}(\sigma, \sigma, 0)$ and $\sigma = (\sigma_1 + \sigma_2)/2$. This matrix is the one closest to the estimated matrix $\mathbf{E_4}$ under the Frobenius norm, that satisfies our requirements. We finally obtain an essential matrix of the form:

$$\hat{\mathbf{E}} = \begin{pmatrix} 0 & \frac{c_1}{\sigma_1} & 0 \\ \frac{c_2}{\sigma_2} & 0 & \frac{c_4}{\sigma_2} \\ 0 & \frac{c_3}{\sigma_1} & 0 \end{pmatrix}. \tag{4.11}$$

As mentioned in Section 4.2 a valid epipolar constraint can be divided into two different components, one rotational component and one skew-symmetric translational one. If we factorize the essential matrix in Equation 4.11 into two such matrices, we get the rotation matrix

$$\mathbf{R} = \begin{pmatrix} c_\beta & 0 & -s_\beta \\ 0 & 1 & 0 \\ s_\beta & 0 & c_\beta \end{pmatrix}, \tag{4.12}$$

with the elements $c_\beta = -(c_1c_2 + c_3c_4)/\sigma_1\sigma_2$ and $s_\beta = (c_1c_4 - c_2c_3)/\sigma_1\sigma_2$. Considering that the determinant equals $c_\beta^2 + s_\beta^2 = 1$, we conclude that $\mathbf{R}$ really is a rotation and that c_β and s_β are the cosine and sine of the vergence angle $\beta = \alpha_l + \alpha_r$ in Figure 4.2. The corresponding translational component

$$\mathbf{S} = \begin{pmatrix} 0 & -s_{\alpha_r} & 0 \\ s_{\alpha_r} & 0 & c_{\alpha_r} \\ 0 & -c_{\alpha_r} & 0 \end{pmatrix}, \tag{4.13}$$

is the second factor of the essential matrix $\hat{\mathbf{E}}$, where the elements $s_{\alpha_r} = c_2/\sigma_2$ and $c_{\alpha_r} = c_4/\sigma_2$ can be identified as $\sin(\alpha_r)$ and $\cos(\alpha_r)$. Thus $\mathbf{S}$ can be interpreted as an outer product of $\mathbf{s} = (\cos(\alpha_r), 0, -\sin(\alpha_r))^\top$, the baseline relative to the right camera frame, and some other vector.

4.4 Parameter estimation

When we estimate the parameters of the different epipolar models, it is expected that a number of extracted corner features is given. As described in Section 4.2, each point in 3D space will be projected onto the camera image planes, producing a pair of image features $(\mathbf{x_l^i}, \mathbf{x_r^i})$, where $\mathbf{x_l^i} = (x_l^i, y_l^i, f_l^i)^\top$ and $\mathbf{x_r^i} = (x_r^i, y_r^i, f_r^i)^\top$. In practice, one could use the Harris corner detector (Harris & Stephens 1988) to locate corners in the left and right images. These features then have to be grouped into correspondence pairs, before the actual epipolar geometry estimation begins. This matching is far from trivial and a good matching is most critical for the performance of the estimation process. A detailed description of the corner matching of the presented system may be found in Section 8.1.

A natural way to recover the epipolar geometry is finding the optimal estimate in a least squares sense, using all the available correspondences. The more data being used, the better the influence of image noise will be suppressed. However, since the errors used in the optimization are squared, even single outliers, that is mismatches of features that do not correspond to the same point in 3D space, will make the estimate useless. How to identify these outliers will be dealt with later on in Section 4.6. As mentioned earlier the epipolar constraint can be estimated using either linear or non-linear methods. Since the essential matrix only has five degrees of freedom, a non-linear approach might be useful. However, since we are interested in computational speed as well as accuracy, we will concentrate on linear methods first and then return to non-linear methods in Section 4.7. A linear method is usually quicker and if accurate enough, we are satisfied.

4.4.1 Orthogonal least squares

Unlike ordinary least squares optimization, where the errors are considered in one coordinate only, an often more successful approach is orthogonal least squares minimization (Torr 1995). For the 8-point method in Section 4.2.1, this will look as follows. If the left and right images vectors are rescaled so that $f_l^i = f_r^i = 1$, a hyper-plane

$$\hat{\mathbf{c}} = \begin{pmatrix} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 \end{pmatrix}^\top, \tag{4.14}$$

defined by the nine parameters of $\mathbf{E_9}$, is fit to a set of sample points given by

$$\mathbf{y_i} = \begin{pmatrix} x_l^i x_r^i & x_l^i y_r^i & x_l^i & y_l^i x_r^i & y_l^i y_r^i & y_l^i & x_r^i & y_r^i & 1 \end{pmatrix}^\top. \tag{4.15}$$

This is done by minimizing the sum of squared orthogonal errors from the points to the hyper-plane, which is determined by the error function

$$f(\mathbf{c}) = \sum_{i=1}^{N} (\mathbf{y_i}^\top \mathbf{c})^2 = \sum_{i=1}^{N} \mathbf{c}^\top \mathbf{y_i} \mathbf{y_i}^\top \mathbf{c} = \mathbf{c}^\top \mathbf{M} \mathbf{c}, \quad \text{where } \mathbf{M} = \sum_{i=1}^{N} \mathbf{y_i} \mathbf{y_i}^\top. \tag{4.16}$$

The trivial case, $\mathbf{c} = 0$, might be avoided using an additional constraint, $\mathbf{c}^\top \mathbf{c} = 1$. The second moment matrix $\mathbf{M}$ is a 9×9 matrix and has eigenvalues λ_k, in increasing order, with corresponding eigenvectors $\mathbf{u_k}$; $k = 1...9$. It is well known that the estimate $\hat{\mathbf{c}}$, that minimizes $f(\mathbf{c})$, can be found as the least eigenvector $\mathbf{u_1}$, that is the eigenvector corresponding to the least eigenvalue λ_1. The total sum of squared errors $f(\hat{\mathbf{c}})$ will then be equal to λ_1 and since $\mathbf{M}$ is symmetric, it will always be non-negative. Once $\hat{\mathbf{c}}$ has been found, it can be rewritten as the matrix $\mathbf{E_9}$. Since there are nine unknowns and $\hat{\mathbf{c}}$ only has to be determined up to scale, eight feature pairs are needed to find the epipolar constraint, which explains why the approach is known as the 8-point method.

If the model $\mathbf{E_4}$ is used instead, the four unknown parameters can be found in a very similar manner. Now a hyper-plane $\hat{\mathbf{c}} = (c_1, c_2, c_3, c_4)^T$ is fitted to the measurement points $\mathbf{y_i} = (x_l^i y_r^i, y_l^i x_r^i, y_r^i, y_l^i)^T$, minimizing $f(\mathbf{c}) = \sum_{i=1}^{N} (\mathbf{y_i}^\top \mathbf{c})^2$. Just as in the case of $\mathbf{E_9}$ this is possible finding the least eigenvector $\mathbf{u_1}$ of $\mathbf{M} = \sum_{i=1}^{N} \mathbf{y_i} \mathbf{y_i}^\top$. However, due to the low dimensionality of $\mathbf{M}$ the process of finding this eigenvector is much simplified.

4.4.2 The Inverse Power method

Speed is of major concern in the proposed system, and it is thus important that computations are made as fast as possible. Since $\mathbf{u_1}$ is the only eigenvector that needs to be determined, one does not have to find all the other eigenvectors. Hence, it is possible to use an iterative approach such as the Inverse Power method (Golub & Van Loan 1996), if convergence is ensured. Initially the adjoint of the matrix $\mathbf{M}$ is calculated, that is

$$\mathbf{M_a} = adj(\mathbf{M}) = det(\mathbf{M}) \cdot \mathbf{M^{-1}}. \tag{4.17}$$

The reason why the adjoint is used, instead of the inverse of $\mathbf{M}$, is simply because the system might be ill-conditioned, which leads to large errors if the adjoint is divided by the determinant when calculating the inverse. Since the final eigenvector will be normalized anyway, this difference in scaling does not change the result. Using an arbitrarily chosen initial vector $\mathbf{v_0}$, a series of updated vectors are calculated iteratively according to

$$\mathbf{v_k} = \frac{\mathbf{M_a v_{k-1}}}{\mid \mathbf{M_a v_{k-1}} \mid}. \tag{4.18}$$

It can be shown that $\mathbf{v_k}$ will converge towards $\mathbf{u_1}$ as $k \to \infty$, and the error will decrease at a rate of λ_1/λ_2. The question is what this actually means in terms of the feature pairs involved.

An analysis of the eigenvalues of $\mathbf{M}$ shows that there is a strong dependency between the scales of the eigenvalues and different aspects of the selected points, such as the level of image noise and how the points are distributed in image space and depth. The smallest eigenvalue λ_1 increases quadratically as the level of noise is doubled, but does not change very much as the distribution of points varies. For the second smallest eigenvalue λ_2 the dependencies are reversed. As the points are spread twice as much over the image space or in depth, the eigenvalue doubles for each dimension. The same eigenvalue is, however, not very sensitive to noise. In conclusion, a low value of λ_1/λ_2, which means rapid convergence, is equivalent to feature points being widely spread in space and suffering from a low level of noise. This property will be used later for eliminating bad estimates.

4.5 Alternative models

There are a number of reasons why the two possible models presented above might not fulfil our expectations under all circumstances. Based on the actual conditions, other models might be taken into consideration. From a computational point of view, there is a large difference in determining the epipolar geometry between an almost parallel camera configuration and a system where the vergence angle is large. The same difference exists depending on the symmetry of the system. A model with too many free parameters, tends to be less robust, even if the match between image data and model is better. This means that if there is any a priori information about the current configuration, that information ought to be exploited, reducing the number of unknown parameters.

Symmetric configurations

If it is known that the camera configuration is symmetric or close to symmetric, it is possible to simplify the model from Section 4.3.1. The reason why this is interesting is the possible improvement in speed. If nothing is sacrificed, a model with as few unknown parameters as possible ought to be used. One such model of the essential matrix looks as follows:

$$\mathbf{E_s} = \begin{pmatrix} 0 & c_1 & 0 \\ c_2 & 0 & c_3 \\ 0 & -c_3 & 0 \end{pmatrix}. \tag{4.19}$$

If $\mathbf{y_i} = (x_l^i y_r^i, y_l^i x_r^i, y_r^i - y_l^i)^\top$, an optimal estimate $\hat{\mathbf{c}} = (c_1, c_2, c_3)^\top$ can be found minimizing $f(\mathbf{c}) = \sum_{i=1}^N (\mathbf{y_i}^\top \mathbf{c})^2$. Similar to the previously presented models, $\hat{\mathbf{c}}$ is given by the least eigenvector of $\mathbf{M} = \sum_{i=1}^N \mathbf{y_i}\mathbf{y_i}^\top$. This is considerably faster than using the model in Section 4.3.1, since $\mathbf{M}$ is only of dimension 3.

It might be possible to reduce the number of unknown parameters even further if the configuration is not just close to symmetric, but also known to have a small vergence angle β. In practice it is rare that β is larger than about 20°. This means that α_l and α_r will be relatively small and the parameter c_3 in the previous model is close to 1. This leads to a model of the form

$$\mathbf{E_p} = \begin{pmatrix} 0 & c_1 & 0 \\ c_2 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}. \tag{4.20}$$

A least squares estimate $\hat{\mathbf{c}} = (c_1, c_2)^\top$ can be found solving

$$\mathbf{A}\hat{\mathbf{c}} = \mathbf{b}, \quad \text{where} \quad \mathbf{A} = \sum_{i=1}^N \mathbf{y_i}\mathbf{y_i}^\top \quad \text{and} \quad \mathbf{b} = \sum_{i=1}^N \mathbf{y_i}(y_r^i - y_r^i), \tag{4.21}$$

given that $\mathbf{y_i} = (x_l^i y_r^i, y_l^i x_r^i)^\top$. Thus the problem is solved without calculating any eigenvectors. This model in a sense represents one extreme of a whole spectrum of models, and it should not be surprising if a model as simple as this only works under very limited conditions.

Image based error models

So far all methods presented have been directed towards minimizing an algebraic error function $|\mathbf{x_l}^\top \mathbf{E} \mathbf{x_r}|$. The main advantage is that it makes a linear solution easy, but it is not always a good choice. As was earlier mentioned in Section 4.2, the epipolar constraint is satisfied if $d(\mathbf{E}) = \mathbf{x_l}^\top \mathbf{E} \mathbf{x_r} = \mathbf{l_l}^\top \mathbf{x_l} = \mathbf{l_r}^\top \mathbf{x_r} = 0$, where $\mathbf{l_l} = \mathbf{E}\mathbf{x_r} = (l_{l,x}, l_{l,y}, l_{l,f})^\top$ and $\mathbf{l_r} = \mathbf{E}^\top \mathbf{x_l} = (l_{r,x}, l_{r,y}, l_{r,f})^\top$ are the epipolar lines in the left and right image respectively. Unlike what might be assumed, $d(\mathbf{E})$ is not equal to the distance between an epipolar line and its corresponding image point, even if this most likely is a desired optimization criteria. The reason is that the epipolar lines have not yet been normalized (Weng et al. 1993).

A question is whether normalization can be performed, without using non-linear optimization. After normalization, the image distance errors in the left and right images are given by

$$d_l(\mathbf{E}) = \frac{\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r}}{\sqrt{l_{l,x}^2 + l_{l,y}^2}} \quad \text{and} \quad d_r(\mathbf{E}) = \frac{\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r}}{\sqrt{l_{r,x}^2 + l_{r,y}^2}} \tag{4.22}$$

respectively. After a few calculations using the true essential matrix given in Section 4.3, it can be shown that $l_{l,x}^2 + l_{l,y}^2 = y_r^2 + x_r^2 \sin^2(\alpha_r) - x_r \sin(2\alpha_r) + \cos^2(\alpha_r)$. A similar equation can be found for the right epipolar line. It is easily seen that if α_r and α_l are small and no extreme focal length is used, the last term dominates, leading to the approximate distance errors

$$d_l'(\mathbf{E}) = \frac{\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r}}{\cos(\alpha_r)} \quad \text{and} \quad d_r'(\mathbf{E}) = \frac{\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r}}{\cos(\alpha_l)}. \tag{4.23}$$

Using $d_l'(\mathbf{E})$, instead of $d(\mathbf{E})$, as the error function means that large α_r are penalized. This leads to a model of the essential matrix, of the form:

$$\mathbf{E_l} = \begin{pmatrix} 0 & c_1 & 0 \\ c_2 & 0 & 1 \\ 0 & c_3 & 0 \end{pmatrix}. \tag{4.24}$$

If $\mathbf{y_i} = (x_l^i y_r^i, y_l^i x_r^i, y_l^i)^\top$ an estimate $\hat{\mathbf{c}} = (c_1, c_2, c_3)^\top$ can be found minimizing $f(\mathbf{c}) = \sum_{i=1}^N (\mathbf{y_i}^\top \mathbf{c} + y_r^i)^2$. The least squares estimate is found solving the equation

$$\mathbf{A}\hat{\mathbf{c}} = \mathbf{b}, \quad \text{where} \quad \mathbf{A} = \sum_{i=1}^N \mathbf{y_i}\mathbf{y_i}^\top \quad \text{and} \quad \mathbf{b} = -\sum_{i=1}^N \mathbf{y_i} y_r^i. \tag{4.25}$$

Using the distance errors in both left and right image would possibly be an even better choice, but just summing the two error functions and solving the problem linearly is not easy. Instead one could use the geometric average of $d_l'(\mathbf{E})$ and $d_r'(\mathbf{E})$. This leads to a new error function

$$d_{lr}'(\mathbf{E}) = \frac{\mathbf{x_l}^\top \mathbf{E}\mathbf{x_r}}{\cos(\alpha_r) + \cos\alpha_l} \tag{4.26}$$

and with this function the problem can be solved linearly. In fact, one would get the same function if the problem was statistically normalized, that is if the variance of the reprojected errors is minimized (Weng et al. 1993). Given the sample points $\mathbf{y_i} = (x_l^i y_r^i, y_l^i x_r^i, y_l^i + y_r^i)^\top$ and the squared error function $f(\mathbf{c}) = \sum_{i=1}^N (\mathbf{y_i}^\top \mathbf{c} + (y_l^i - y_r^i))^2$, an estimate can be found solving

$$\mathbf{A}\hat{\mathbf{c}} = \mathbf{b}, \quad \text{where} \quad \mathbf{A} = \sum_{i=1}^N \mathbf{y_i}\mathbf{y_i}^\top \quad \text{and} \quad \mathbf{b} = \sum_{i=1}^N \mathbf{y_i}(y_r^i - y_l^i). \tag{4.27}$$

The final estimate of the essential matrix will be given by

$$\mathbf{E_n} = \begin{pmatrix} 0 & c_1 & 0 \\ c_2 & 0 & c_3 + 1 \\ 0 & c_3 - 1 & 0 \end{pmatrix}. \tag{4.28}$$

Common to each model presented in this section, is that once an estimate has been found, a projection down to the two-dimensional space of possible essential matrices has to be made. Similar to the ordinary constrained case, this is done creating a new matrix $\hat{\mathbf{E}}$ as in Equation 4.11.

4.6 Robust estimation

For the estimation of the epipolar geometry we have chosen corner features. This is because they do not suffer from the aperture problem and their positions can be determined in both image dimensions. However, since the corners are viewed from two different angles by two different physical cameras, they might look slightly different. If transparencies and specularities exist in the scene, the matching procedure is further complicated. It is also possible that due to occlusions some 3D points may only be seen by one of the cameras and the corresponding image features cannot be matched at all. The matching of image features is typically performed through correlation of image pixel windows centred on the features. In the case of occlusions, a large fraction of these pixels may belong to the background that differs between the two camera images, leading to additional complications. A final reason why erroneous pairs can be expected is that repetitive patterns exist in the scene and many points look much the same.

It is essential that these outliers can be identified and eliminated from the data set, especially if an optimum is found solving a least squares system, where the outliers would otherwise dominate the result. The outliers should be detected as early as possible. To determine a final estimate all the remaining, and hopefully correct, image features could be used. This can be done by selecting a small set $M(j)$ of random feature correspondences and calculating an essential matrix as described above. The procedure is repeated numerous times and for each set the estimated epipolar geometry is evaluated. Using some kind of error measure, estimates with large errors could then be ignored. Hopefully, estimates based on feature sets that include one or more outliers will belong to these.

A commonly used method known as the random sampling consensus paradigm (RANSAC) (Fischler & Bolles 1981, Torr & Murray 1997) uses the complete set of correspondences in order to test the epipolar estimates. Once an estimate has been found the image distance error, that is the distance between a feature and its corresponding epipolar line, is calculated for each and every feature pair. If this error is lower than a certain threshold, the epipolar estimate is said to be supported by the feature pair. The estimate supported by most feature pairs is then selected as the best estimate of the true epipolar geometry. However, since all image correspondences are used for this evaluation, RANSAC might be too time-consuming and damage the performance of the complete system.

4.6.1 Quality testing estimates

In order to decide whether an estimate should be ignored or not, the least eigenvalue, that is the sum of squared errors, could be analyzed. This can be done implicitly, observing the convergence of the iterative power method, as was described in Section 4.4.2. Unfortunately, this does not guarantee that the estimate is actually a good estimate located close to the true epipolar geometry. Figure 4.3 shows a number of sample points found when estimating the epipolar geometry according to this scheme. Further experiments will be presented in more detail later in Section 4.8. As seen in the figure, the points are clustered around more than one centre in the two-dimensional space of epipolar geometries. The cluster near $(7, 3)$ corresponds to the true epipolar geometry.

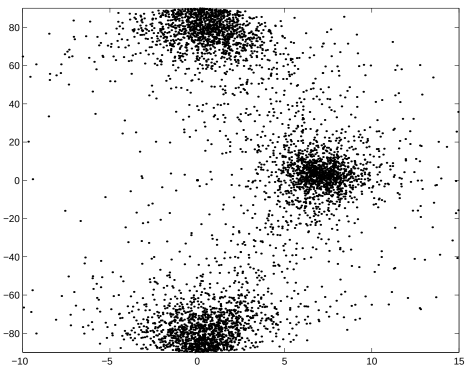

Figure 4.3. Distribution of sample points when estimating the epipolar geometry. The x-axis represents the vergence angle and the y-axis the gaze direction. The data set consists of 500 feature pairs, including 20% outliers.

The existence of the two other clusters can be understood as follows. If the selected feature points suffer from a high level of noise or if the feature set includes one or more outliers, the two eigenvectors $\mathbf{u_1}$ and $\mathbf{u_2}$ may change order. This problem has earlier been described by Ma et al. (2001). If the eigenvalues change order, the wrong eigenvector represented by the two other cluster centres in Figure 4.3 will be found instead. Figure 4.4 shows the epipolar lines associated with solutions from the two different clusters. Both images are based on the same set of feature pairs, but the right one includes a point that has an additional error of 10 pixels in the y-position. The sensitivity may be explained as follows. If the cameras are fixated on a point in the centre of the feature cloud and the points are poorly spread in depth, the image displacements are typically small. A relatively small error at one point may then suddenly dominate, which may lead to the incorrect solution being found.

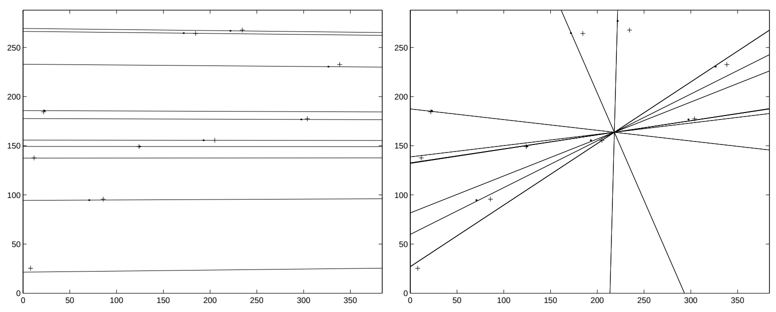

Figure 4.4. Epipolar lines for two solutions based on 10 feature points. The right image corresponds to the incorrect eigenvector, which is the result of a 10 pixel error at one point.

In order to separate such false samples from real ones, one may employ two additional quality tests to decide whether to accept a sample point as a good estimate of the epipolar geometry. The first test ensures that the maximum image distance error is reasonably small, that is the errors of Equation 4.16 have to be lower than some predefined threshold, which could be set to a value similar to the variance of the feature extractor. The other test involves a check whether every point in $M(j)$ is actually located in front of the cameras, when the three-dimensional positions have been reconstructed.

The left and right optical axes seldom coincide in a single 3D point. This is because there will always be some noise in the system and the calculations cannot be done totally free from small truncation errors. However, here this fact will simply be ignored, since the y-coordinate of the reconstruction does not affect the test. With the coordinate system centred at the left camera, the two projections will be given by $\mathbf{x_l} = f_l^{-1}\mathbf{R_l}\mathbf{u} = (x_l, y_l, 1)^\top$ and $\mathbf{x_r} = f_r^{-1}\mathbf{R_r}(\mathbf{u} - \mathbf{t}) = (x_r, y_r, 1)^\top$, following the derivations in Sections 4.2 and 4.3. Eliminating the 3D position $\mathbf{u}$ and dropping the y-coordinates, the distances, f_r and f_l, from $\mathbf{u}$ to the two cameras can be found solving

$$\begin{pmatrix} x_r & -\cos(\beta)x_l - \sin(\beta) \\ 1 & \sin(\beta)x_l - \cos(\beta) \end{pmatrix} \begin{pmatrix} f_r \\ f_l \end{pmatrix} = \begin{pmatrix} \cos(\alpha_r) \\ \sin(\alpha_r) \end{pmatrix}.$$

In order for the in-front test to be successful both distances have to be positive. After performing the two quality tests on the sample points in Figure 4.3 and omitting the points that do not pass, only one cluster will remain, as shown in Figure 4.5.

4.7 Non-linear optimization

The linear method described in Section 4.4 relies on a minimum found in a space of four dimensions. However, what is really sought is the minimum of the

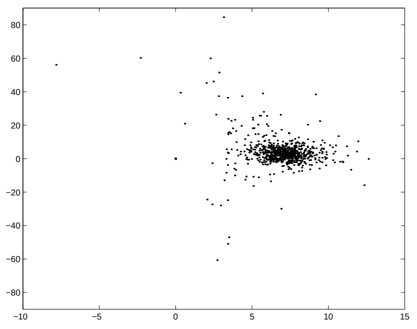

Figure 4.5. Distribution of sample points, after filtering out samples that do not pass the two quality tests. The simulation was performed in the same manner as for the points in Figure 4.3.

objective function on a two-dimensional manifold in this space. A point on the manifold is found by performing a projection, as was described in Section 4.3.1. The problem with this approach is that one does not know whether this point really lies close to a local minimum.

A non-linear approach could be used to ensure that this is the case. Given the two angles $\alpha = (\alpha_l, \alpha_r)$ the corresponding point in the four-dimensional space is given by

$$\mathbf{f}(\alpha) = (-\sin(\alpha_l), -\sin(\alpha_r), \cos(\alpha_r), -\cos(\alpha_l))^T. \tag{4.29}$$

In order to find the minimum of the objective function

$$E(\alpha) = \frac{1}{2}\mathbf{f}(\alpha)^T\mathbf{M}\mathbf{f}(\alpha), \tag{4.30}$$

it is possible to use Newton's method. The minimum is found iteratively, in each iteration using a direction $\mathbf{p_k}$ that satisfies

$$\nabla^2 E(\alpha^{(k)})\mathbf{p_k} = -\nabla E(\alpha^{(k)}). \tag{4.31}$$

As an initial estimate $\alpha^{(0)}$, one may use a point from the linear optimization explained earlier. Since the function $E(\alpha)$ typically is well-behaved around the minimum, the estimate can be updated without any additional backtracking operation. Hence the *(k+1)* th estimate of α is

$$\alpha^{(k+1)} = \alpha^{(k)} + \mathbf{p_k}. \tag{4.32}$$

The gradient and Hessian of the optimization function are given by

$$\nabla E(\alpha) = (\mathbf{f^T M}\frac{\partial \mathbf{f}}{\partial \alpha_\mathbf{l}}, \mathbf{f^T M}\frac{\partial \mathbf{f}}{\partial \alpha_\mathbf{r}})^T \text{ and} \tag{4.33}$$

$$\nabla^2 E(\alpha) = \begin{pmatrix} \mathbf{f}^{\mathbf{T}}\mathbf{M}\frac{\partial^2 \mathbf{f}}{\partial \alpha_{\mathbf{l}}^2} & 0 \\ 0 & \mathbf{f}^{\mathbf{T}}\mathbf{M}\frac{\partial^2 \mathbf{f}}{\partial \alpha_{\mathbf{r}}^2} \end{pmatrix} + \begin{pmatrix} \frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{l}}}^{\mathbf{T}}\mathbf{M}\frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{l}}} & \frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{l}}}^{\mathbf{T}}\mathbf{M}\frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{r}}} \\ \frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{r}}}^{\mathbf{T}}\mathbf{M}\frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{l}}} & \frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{r}}}^{\mathbf{T}}\mathbf{M}\frac{\partial \mathbf{f}}{\partial \alpha_{\mathbf{r}}} \end{pmatrix}.$$

Since non-linear optimization is relatively costly, it is important to discard bad estimates as soon as possible. In our implementation the sum of squared errors is tested directly after performing the initial linear optimization. This turns out to be a quick way of filtering out most sets with image correspondences that include one or more outliers, before continuing with the non-linear optimization. The convergence test of the Inverse Power procedure described earlier ensures that the function $E(\alpha)$ is well-behaved and few iterations are required for the non-linear optimization. As seen in Figure 4.6 the spread of sample points is significantly reduced, especially in the dimension of the gaze direction.

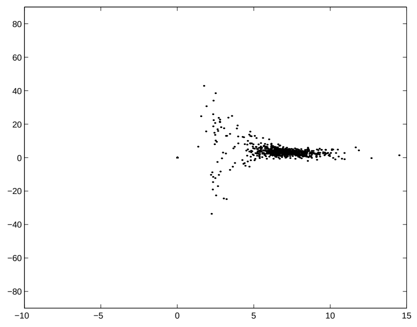

Figure 4.6. Distribution of sample points, after non-linear optimization and filtering out samples of low quality. The simulation was performed in the same manner as for the points in Figure 4.3.

4.8 Experiments

To evaluate the alternative methods, a number of simulations were performed. Series of 500 randomly generated points were evenly spread in a three-dimensional truncated pyramid located around the fixation point, that is the point where the two optical axes intersect. With a focal length f equivalent to about 400 pixels, based on 360×288 pixel image planes, the approximate distance to the fixation point is $Z_f = 2f/\beta$, where β is the vergence angle. The generated points were spread between $0.5Z_f$ and $2Z_f$ from the observer. Each point was projected onto the image planes of the left and right cameras. Noise with a standard deviation of one pixel was added to each image dimension, in order to simulate feature extraction errors. The effect of outliers in real data was tested, adding an additional 30 pixel noise source to the left feature point position of about 20% of the 500 feature pairs.

The first approach that was tested is based on linear optimization (**linear**) using the $\mathbf{E_4}$ model of the essential matrix, as described in Section 4.3.1. A total of 10000 different sets, $M(j)$, consisting of six image correspondences each were generated with random sampling. For each set the corresponding essential matrix was calculated. The quality of the results was then tested and bad estimates discarded. Typically only about 10% of the estimates are left after this procedure, which might seem a low figure. However, based on the number of outliers, it can be expected that about $1-(1-0.20)^6 = 74\%$ include at least one outlier and these estimates will thus be excluded. Additional estimates are thrown away because the feature points are not properly spread in 3D space.

The remaining epipolar estimates may be described by sample points in a two-dimensional space, with vergence angle and gaze direction as coordinates. The points typically form an ellipsoidal cluster in this space, which can be seen in Figure 4.5. The mean sample position, the centroid, can then be used as the final estimate of the epipolar geometry. The size and shape of the ellipsoid varies depending on the level of image noise and the true camera configuration. As explained earlier, linear optimization does not necessarily provide an optimum in this two-dimensional domain. In order to ensure that a local minimum really is found, one may add an extra nonlinear optimization operation (**nonlinear**). This approach was tested using the same randomly generated sets of points and compared with the linear method.

Instead of just using the mean sample point position as the final estimate of the epipolar constraint, one may also test its support among the complete set of image correspondence. In this approach (**support**), the epipolar estimate with the highest number of supporting feature pairs wins and will thus be used as the final estimate. The estimate $\hat{\mathbf{E}}$ is supported by a pair of points $(\mathbf{x_l}, \mathbf{x_r})$, if $|\mathbf{x_l^T}\hat{\mathbf{E}}\mathbf{x_r}| < \sigma_E$. The constant σ_E is a predefined threshold related to the expected noise of the corner extractor.

There are primarily two reasons for keeping the degrees of freedom of the binocular system as few as possible. The first reason is the computational speed and the other is the robustness. In order to test these properties, the previous three approaches, that were all based on $\mathbf{E_4}$, were compared to a method using the full set of five degrees of freedom. This method, which uses the $\mathbf{E_9}$ essential matrix model from Section 4.2.1, is a Least Median of Squares (**LMedS**) approach proposed by Zhang et al. (1995). For each randomly sampled set, an epipolar geometry estimate $\mathbf{E_{LMedS}}$ is calculated and for each feature pair in the whole data set the squared error is obtained. The median of the squared errors then determines the quality of the set. The feature set that leads to the lowest median error is considered the winner and determines the final estimate.

In order to compare the different methods, the results of the **LMedS** approach had to be converted into two parameters only, the vergence angle and gaze direction. First the essential matrix $\mathbf{E_{LMedS}}$ was decomposed into a rotational and a skew-symmetric matrix. The two matrices of the forms given by Equations 4.12 and 4.13, that were closest to those of the estimation process, under the Frobenius norm, were then determined. The corresponding two parameters were then used in the evaluation.

Experimental results

Simulations were performed using the methods mentioned above. The resulting vergence angles can be seen in Figure 4.7, which show the standard deviation and mean of the estimated angles. Different groups of bars illustrate the results from different combinations of true vergence angles and gaze direction. Vergence angles of 2°, 7° and 17° represent the minimum, typical and maximum values of a stereo head system under normal working conditions. The thin horizontal lines in the right image of Figure 4.7 show the true vergence angles and can be used as references.

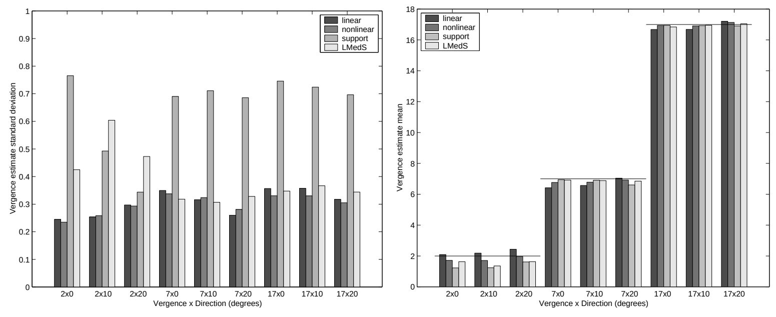

Figure 4.7. Standard deviation (left) and mean (right) of the vergence angle, for different combinations of true vergence angles and gaze directions.

We see that all methods come close to the true configuration, since the mean error never exceeds one degree and the standard deviations are reasonable small. In general the **support** method is not as robust as the other methods. It is also worth noting that, contrary to what the higher number of degrees of freedom might suggest, the **LMedS** method is just as robust as the first two methods, unless the vergence angle is too small. We may also note a slight improvement of the bias when nonlinear optimization has been added to the ordinary linear one, but it is questionable if the relatively small improvements justify the increased complexity of the method.

Considering the graphs in Figure 4.8 that show the mean and standard deviation of the gaze direction, one may conclude that this component is much harder to estimate. Here the methods based on the $\mathbf{E_4}$ model show a much better robustness than the **LMedS** approach. Systems where the cameras are close to parallel are especially hard and even the best methods show a radically reduced robustness. However, a standard deviation of about three degrees, which is equivalent to the worst results of the **linear** and **nonlinear** methods, is probably acceptable in most applications. The method based on **support** does not come close to the robustness of the first two methods, even if the results are considerably more stable than those of **LMedS**.

In conclusion, the gaze direction is far easier estimated using the **linear** and **nonlinear** methods than with the methods based on a support measure. However, for the vergence angle all methods show similar results and there is no obvious factor excluding one method from the others. In most cases, the **nonlinear** method is slightly less biased than the **linear** method, especially for the gaze direction. It is not always necessary to test as many as 10000 sets of six feature pairs each. More sets are typically required when the cameras are close to parallel due to the essential matrix not being of full rank. It should be pointed out that the support based methods are more sensitive to a reduced number of tested feature sets, than methods based on the centroid of accepted estimates. Later in Section 4.11.1 the methods will be compared in terms of the computational cost.

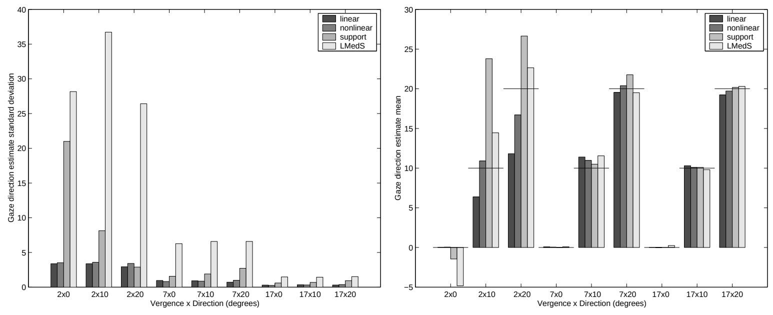

Figure 4.8. Standard deviation (left) and mean (right) of the gaze direction, for different combinations of true vergence angles and gaze directions.

4.9 An iterative algorithm

So far all experiments have been based on the epipolar geometry using essential matrices. As described in Section 4.2, the epipolar constraint decouples the structure from the stereo problem. Without knowing anything about the structure, the relative location and orientation of the cameras can be extracted. However, in many applications the depths are in fact known, at least approximately, and the question is whether this knowledge can be used to simplify the problem. One might not have an accurate depth value for each individual image point, but at least the expected distribution of depths may be given.

Instead of using the essential matrix, one may view the problem as a differential motion problem. The left and right images can be considered as two images taken by the same cameras at two different instances in time, while the camera moves in 3D space. The vertical and horizontal disparities can be used as approximation of the optical flow. One might wonder whether such a treatment of the problem is feasible, since such approximations are known to break down if the motion from frame to frame is too large, and we intend to use the model for stereo rather than for motion. However, in this section it will be shown that for a stereo head in fixation, where objects in space are located relatively far away from the platform and the vergence angles are small, one will stay within the bounds of acceptable results.

Considering a stereo head as shown in Section 4.2 and the bilinear optical flow equation of Longuet-Higgins & Prazdny (1980), the difference in position between an image point (x_i, y_i) in the left and right image is approximately

$$\begin{pmatrix} dx_i \\ dy_i \end{pmatrix} = \begin{pmatrix} 1 + x_i^2 \\ x_i y_i \end{pmatrix} \beta + \frac{1}{Z_i} \begin{pmatrix} T_x - x_i T_z \\ -y_i T_z \end{pmatrix}. \tag{4.34}$$

The values dx_i and dy_i represent the horizontal and vertical image displacements. Unlike with the essential matrix, the depth Z_i of each point appears in the equation. The angle β represents a rotation around the y-axis, as seen in Figure 4.2, which is the same as the vergence angle. The gaze direction is determined by the translational component given by T_x and T_z, that is the position of the right camera relative to the left.

Since the camera system is constrained, such as described in Section 4.3.1, the horizontal disparities of different points maintain the same ordering if the cameras are verge, that is change the vergence angle (Yau & Wang 1999). This means that if the distribution of depths is known in advance, one may easily find an approximate depth for each individual image point. This will be shown in the algorithm outlined below.

The approach proposed in this section is a two-stage iterative algorithm that alternates between estimating depths and the camera configuration. First the geometry parameters β, T_x and T_z are calculated using the current depth estimates. In the second stage the depth values Z_i are updated from the new geometry parameters. The process is initiated using an estimated vergence angle calculated from an expected median depth and the median of horizontal disparities. As seen in Equation 4.34, the depths and translation are related and can only be estimated up to a constant. Letting $t = T_z/T_x$ and denoting the inverted and scaled depth $\lambda_i = T_x/Z_i$, the equation may be rewritten as

$$\begin{pmatrix} dx_i - \lambda_i \\ dy_i \end{pmatrix} = \begin{pmatrix} 1 + x_i^2 & -x_i\lambda_i \\ x_i y_i & -y_i\lambda_i \end{pmatrix} \begin{pmatrix} \beta \\ t \end{pmatrix}. \tag{4.35}$$

An **iterative** algorithm based on this equation can be seen in Figure 4.9.

4.9.1 Reducing the effect of outliers

As mentioned earlier in Section 4.6, one always has to expect that a considerable fraction of matched feature pairs are in fact outliers, that is feature points

Algorithm

1. Initialize $\beta^{(0)}$ from the expected median distance to points in 3D space and the median horizontal disparity, and let $t^{(0)} = 0$.
2. Individually update the inverted depths using the estimated of the epipolar geometry.

$$\lambda_i^{(n)} = \frac{\mathbf{e_i^T b_i}}{\mathbf{e_i^T e_i}}, \quad \text{where} \tag{4.36}$$

$$\mathbf{b_i} = \begin{pmatrix} dx_i - (1+x_i^2)\beta^{(n)} \\ dy_i - x_i y_i \beta^{(n)} \end{pmatrix} \quad \text{and} \quad \mathbf{e_i} = \begin{pmatrix} 1 - x_i t^{(n)} \\ -y_i t^{(n)} \end{pmatrix}. \tag{4.37}$$

3. Calculate the geometry parameters from the estimated depths, in a least squares framework with all valid feature pairs included:

$$\begin{pmatrix} \beta^{(n)} \\ t^{(n)} \end{pmatrix} = (\sum_i \mathbf{A_i^T A_i})^{-1} \sum_i \mathbf{A_i^T d_i}, \quad \text{where} \tag{4.38}$$

$$\mathbf{A_i} = \begin{pmatrix} 1+x_i^2 & -x_i\lambda_i^{(n-1)} \\ x_i y_i & -y_i \lambda_i^{(n-1)} \end{pmatrix} \quad \text{and} \quad \mathbf{d_i} = \begin{pmatrix} dx_i - \lambda_i^{(n-1)} \\ dy_i \end{pmatrix}. \tag{4.39}$$

4. Return to 2. until convergence is reached.

Figure 4.9. An iterative epipolar geometry estimation algorithm

that do not correspond to any existing points in 3D space. Similar to methods based on the essential matrix, outliers may seriously damage the performance of the algorithm and a couple of adjustments have to be made to improve the robustness.

The most obvious and dominating outliers can be identified by making sure that the vertical disparities are kept within a limit defined by

$$|dy_i| \le |x_i y_i| \max(\beta) + |y_i| \max(T_z/Z_i) + \sigma_y. \tag{4.40}$$

The constant $\max(\beta)$ defines the maximum vergence angle possible, which typically has a value of about 20°. The maximum translation along the optical axis, typically only a few centimetres, relative to the closest object in 3D space is determined by $\max(T_z/Z_i)$, whereas σ_y is the expected noise of the feature extractor. Image correspondences that do not pass this test will be omitted from the data set and not used in the optimization. The test can either be performed before Step 1 or directly after. If the test is done afterwards it is possible to take advantage of the initial vergence estimate $\beta^{(0)}$, which is usually within a few degrees from the true value. A more conservative limit may then be used, allowing also less critical outliers to be identified.

In order to further suppress the effect of outliers, the current implementation assigns a weight w_i to each correspondence and uses these weights in the least squares optimization of Step 3. The errors of Step 2, $\varepsilon_i = \mathbf{d_i} - \lambda_i^{(n)}\mathbf{e_i}$, define the weights. The weights reflect in what sense the horizontal and vertical disparities can agree on a single depth value and determine how much a certain feature influences the final solution. Letting σ_e be a predefined value relating to the noise of the feature extractor, one might use the following definition of weights:

$$w_i = \left\{ \begin{array}{ll} 1 - (\frac{\varepsilon_i}{\sigma_e})^2, & \text{if } |\varepsilon_i| < \sigma_e \\ 0, & \text{otherwise} \end{array} \right\}. \tag{4.41}$$

After introducing the weights given above, Equation 4.38 is replaced by

$$\begin{pmatrix} \beta^{(n)} \\ t^{(n)} \end{pmatrix} = (\sum_i w_i \mathbf{A_i^T A_i})^{-1} \sum_i w_i \mathbf{A_i^T d_i}, \tag{4.42}$$

which leads to a new epipolar estimate, where points that do not match the current estimate are suppressed, thus reducing the effect of outliers.

It should be noted that the success of this procedure depends on the camera configuration being within bounds previously defined. With unusual configurations or when the initial depths values are far away from the true values, correct feature correspondences may be eliminated erroneously and outliers kept unnoticed. The effect of this will be seen in the next section.

4.9.2 Simulations

Simulations similar to those presented in Section 4.8 were carried out using the same sets of randomly generated feature points. In order to test the effect of incorrect initial median depths, the fraction between the expected median depth and the true one was varied between 0.2 and 5.0. The different groups of bars in Figure 4.10 represent estimated vergence angles for different combinations of true vergence angle and gaze direction. The $\mathbf{E_4}$ model was used for the results based on the essential matrix. As seen in the left image, the standard deviation is considerably lower than that of earlier presented methods, provided the expected depth is close enough to the true one. However, when the initial guess was not as good, the algorithm collapses for combinations of large vergence angles and asymmetric camera settings.

Gaze directions estimated using the proposed iterative approach also seem to be more robust than previous methods, as illustrated by Figure 4.11. Even if small vergence angles are still critical, the standard deviations stay within about 2°, which ought to be acceptable in most cases. Unlike the earlier methods, the robustness is slightly dependent on the true gaze direction. This is likely to be a result of the initial translation set in Step 1 of the algorithm.

The iterations converge at a rate remarkably independent of the true camera configuration. Typically 20 to 30 iterations are needed for the relative errors

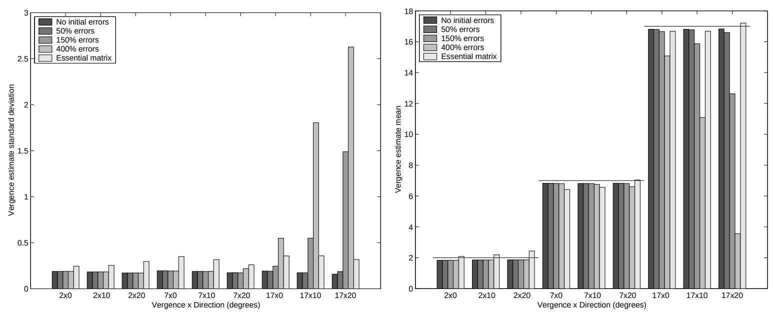

Figure 4.10. Standard deviation (left) and mean (right) of the vergence angle, for different combinations of true vergence angle and gaze direction.

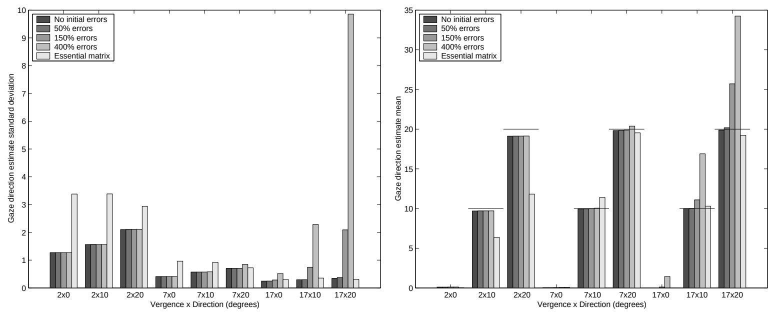

Figure 4.11. Standard deviation (left) and mean (right) of the gaze direction, for different combinations of true vergence angle and gaze direction.

of the depth values to decrease to about 10% of the initial errors from Step 1. The rate is slightly higher for near-symmetric camera configurations and small vergence angles. In situations where outliers have not been properly eliminated the convergence rate can be much worse, even if the procedure never seems to diverge. Based on these observations it might be possible to improve the speed of convergence using forward prediction, but this was not used here.

4.10 Solving for additional parameters

In the previous sections the epipolar geometry was only determined with two degrees of freedom. The question arises of whether more parameters ought to be added to the optimization problem. There were primarily two reasons for only two to be included. The first one is the computational speed. The more often the epipolar geometry estimate can be updated, the easier it will be to take advantage of temporal consistency. Secondly, too many parameters added to the problem might seriously affect the robustness. Oliensis & Govindu (1999) have shown that an optimization with internal camera parameters treated as unknowns, does not necessarily result in more robust results than when the "known" internal parameters are in fact only approximate. In fact, if a parameter is hard to determine, it often means that a small error in this parameter will not significantly effect the estimation of the other parameters.

Hartley (1992) has previously shown that the two focal lengths may be found by exploiting all the degrees of freedom of $\mathbf{E_9}$. However, in our particular case, with the system constrained as is Section 4.3.1, this is not possible (Brooks et al. 1996). The reason is that $\mathbf{E_4}$ always consists of five elements equal to zero. With only three degrees of freedom, it is not possible to estimate both focal lengths simultaneously. However, if the focal lengths are known to be equal, they may be determined, but at the cost of a considerably higher variance in the estimated parameters (de Agapito et al. 1998). It can be shown that the errors may be extremely large if the camera configurations are close to symmetric, which is often the case for binocular stereo heads. As a consequence of this, the presented system does not include such a process. In fact, a moderately erroneous focal length does not hurt the robustness to a great extent, but the results will be slightly biased. More critical to the robustness is the relative difference in focal lengths between the two cameras. This difference can be robustly estimated in a separate operation, simply by assuming that one camera has a known focal length and then search the other. Even if the given focal length is just approximately known, the calculated relative difference will be close to the true value.

For a typical stereo head system, it is sometimes hard to fulfil the requirement that there is no relative tilt between the two cameras. Due to the mounting of the cameras and the placement of the actual CCD chips inside the cameras, there can easily be a bias in vertical disparity of as much as 10 pixels. Without adjusting the extracted feature positions for such a bias, the final estimated geometry will either be useless or seriously biased. The presented system uses the fact that no vertical disparities should be present along the x-axis. As an initialization, features are extracted from a sequence of pairwise images. Using feature correspondences located along the x-axis, a linear relation between the x-positions and the vertical disparities is found. The data are fitted to a straight line, with least median of squares. From this relation the vertical difference in position may be found in the centre of the image. If the epipolar geometry is available from a previous frame, rectified data may be used instead. Through the same procedure, feature positions can also be adjusted to correct for small relative rotations around the optical axes.

4.11 Real-time experiments

A series of real-time experiments, out of which one is presented here in greater detail, was also performed using a binocular stereo head with processing done on an SGI Octane machine. The tested system consists of three basic components, feature extraction, matching and epipolar geometry estimation. Corner features are extracted using the Harris corner detector (Harris & Stephens 1988). In order

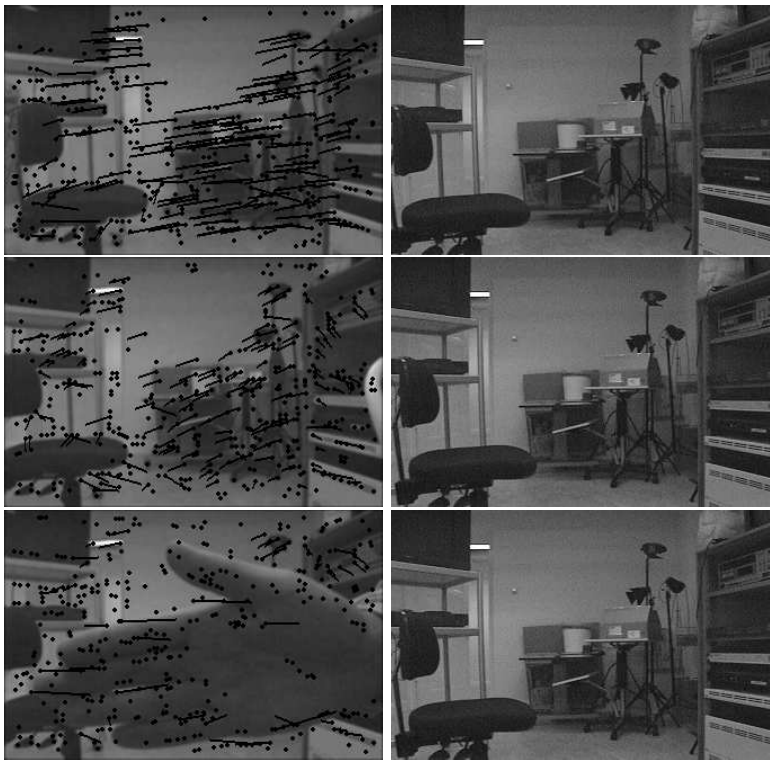

Figure 4.12. Left and right camera images from the 50th, 100th and 150th image frames. Black points represent the extracted corner features, while lines indicate valid matches between the two images.

to suppress image noise, incoming images are low-pass filtered before corners are extracted. Matching is done from the left to right images and then in the opposite direction. With modified normalized cross-correlation of 9×9 pixel areas, a favourite in the opposite image is found for each corner feature. Two features that are each others favourites are considered as reliable and are then used in epipolar estimation process. See Section 8.2 for more details.

The epipolar geometry is determined using a combination of the iterative algorithm presented in Section 4.9 and the method based on the $\mathbf{E_4}$ model of the essential matrix, as described in 4.3.1. First the iterative method is used and if the estimated vergence angle is larger than $12°$, the essential matrix is applied instead. Thus the inherent limitations of the optical flow constraint can be overcome, without reducing the robustness. In the presented sequence the stereo head was kept static, but with the ability for the cameras to verge. On a 195 MHz SGI Octane the complete system runs at 12.5 Hz, which includes grabbing of image frames and on-screen output of left and right images.

A small number of images from the sequence are presented in Figure 4.12, showing images from the left and right cameras in the left and right columns respectively. The three rows constitute image pairs from the 50th, 100th and 150th frames. Between the first and second rows the cameras verge from about 8° to 11.5°. In order to test the ability of the system to reassume convergence after failure, we let someone block the left camera with a hand for a few seconds. This can be seen in the images from the 150th frame. For the left images the locations of extracted corner features are shown, together with lines that indicate valid matches. The system was able to compute the relative tilt between the cameras as 5.2 pixel, using the method presented in Section 4.10.

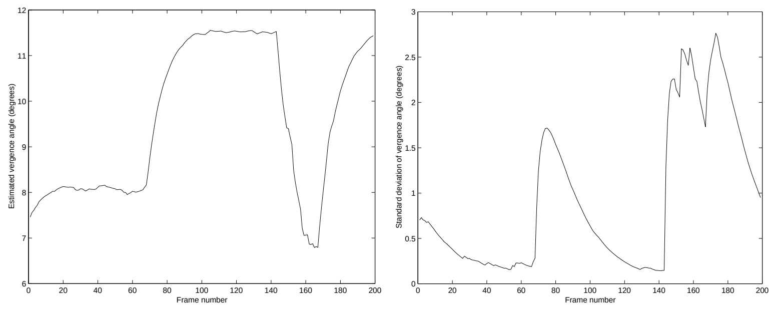

Figure 4.13. Estimated vergence angle (left) and its standard deviation (right). Note the different scales.

The two diagrams of Figure 4.13 show the estimated vergence angles and their standard deviations. The values are processed through a moving average filter, with a time window of about 20 frames. The first frame is the very moment when the system starts running. In Figure 4.14 the corresponding graphs of the estimated gaze directions are shown. The standard deviations stabilize at about 1°, which is close to the errors reported from the simulations in Figure 4.11. The same is true for the vergence angles, where the standard deviation reaches about 0.2° after convergence.

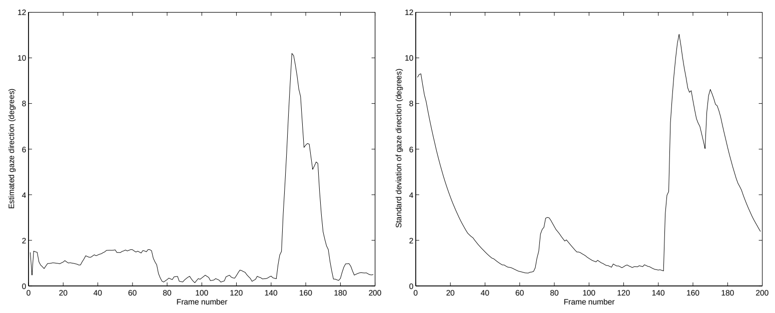

Figure 4.14. Estimated gaze direction (left) and its standard deviation (right).

Method	Cost on R10K
Linear	58 ms (26 ms)
Nonlinear	81 ms (36 ms)
Support	480 ms
LMedS	850 ms
Iterative	27 ms (12 ms)

Figure 4.15. Computational costs of the different methods performed on a 195MHz MIPS R10K processor.

Through all experiments, the bootstrapping process never seemed to fail, provided the distribution of features was otherwise good enough. However, if the lighting-conditions radically change or the cameras get partially occluded as in the 150th frame of the sequence, one might get serious errors primarily due to a dramatic increase in the number of outliers. In order to minimize the effect of such problems, a gate was introduced just before the moving average filter. This gate simply discards inputs to the filter if the estimated values are too far away from the range of values that are considered likely. The range is based on the maximum acceleration of the stereo head and is never changed during execution. For the results presented here such a gate was never applied.

4.11.1 Computational costs

The table in Figure 4.15 shows the computational costs of the methods presented above, when executed on a 195 MHz MIPS R10K processor. The time required for feature extraction has not been included. The data clearly show that methods that do not rely on any support calculations run considerably faster than if a support measure has to be obtained for each and every sample set. It is worth noting that **LMedS** is not much slower than **support**, even if the number of degrees of freedom is six, instead of two. This indicates that the actual support calculations, which are similar in both methods, dominate the total computational cost for these methods.

As seen in the table, the **linear** and **nonlinear** are about an order of magnitude faster. However, it is hard to tell which of the two methods one would prefer. In applications where the slight bias of the **linear** method is not critical, one might benefit from a somewhat lower computational cost. The speed of the two methods may be improved, if a limit is set on the number of epipolar estimates one wishes to include when calculating the centroid. Instead of generating as many as 10000 random sets of six feature pairs each, one might be satisfied when a certain number of accepted estimates have been found and then end the random sampling procedure. The data in brackets show the execution times when such a limit has been set to 200. The accuracy due to such a limit is not notable. The disadvantage is the fact that the computational cost will be less deterministic, but the simulations show that as long as the vergence angle exceeds about 3°, the execution times will not vary very much.

The **iterative** approach turned out to be considerably faster than all the other methods. This is basically because the approximate distance to objects in 3D space is already known and this knowledge has then been used to improve the estimation process. In cases where the external calibration is performed continuously and the camera configuration has not radically changed since the last update, one might benefit from the use of previously estimated camera parameters. Instead of using as many as 25 iterations, the method often converges after much fewer iterations when the previous estimate has been used for initialization. The iterative procedure might then be terminated as soon as convergence is met. At the cost of a less deterministic computational cost, the average execution time will then be 12 ms, as shown in brackets.

4.12 Conclusions

For an active observer, stereo is an important cue for a variety of visually guided tasks, such as navigation, obstacle avoidance and manipulation. However, different tasks require observations of objects located at different depths and because of the limitations in field of view and image resolution, dynamic vergence typically becomes a necessity. Dynamic fixation may also be used to simplify problems like ego-motion estimation and figure-ground segmentation.

In this section a real-time framework for epipolar geometry estimation has been presented, with experiments performed on a binocular head. Instead of using counters on the motors controlling the motion of the stereo head, external calibration is performed continuously at a low computational cost. Two models have been explored, with the number of unknown parameters minimized without affecting the flexibility of the system. The first model based on the essential matrix proved to be somewhat sensitive to near-parallel camera systems, but for vergence angles larger than about $5°$ the errors stay within about $1°$ in both translational and rotational component.

An alternative approach based on the bilinear optical flow equation was also presented. Using an initial expectation of the median distance to objects in 3D space, the geometry parameters and depths were sought in an iterative procedure. Since the approximate model that was used only works for small rotations and translations, it could be expected not to be feasible for modeling the image displacements of a stereo system. However, this study showed that it could in fact be used for a stereo head under normal working conditions. The method was faster and also more robust, especially for small vergence angles, than methods based on the essential matrix.

Chapter 5

Binocular disparities

There are a number of possible cues to perception of depth, such as motion parallax, accommodation, relative size, texture gradients and shadows, but binocular disparities are likely to be the strongest ones. For an autonomous observer, the ability to segregate the world into regions of different depths is crucial. But at long ranges in navigation and short ranges in manipulation binocular disparities form a valuable piece of information. However, the use of multiple cameras introduces some new questions that may be hard to solve. The increased complexity may slow down the system, as data to be analyzed grows. Unless the cameras have been calibrated, intrinsically and externally, the calculated data are difficult to use. Most disparity methods rely on cameras being aligned, such that their epipolar lines are parallel. Thus in the case of verging camera systems, images first have to be rectified, before disparities can be estimated. These problems have limited the use of binocular disparities in robot vision. Nevertheless, we believe that they can be overcome and that binocular stereo should be used by a "seeing" system.

In Chapter 4 the epipolar geometry of a binocular stereo system was analyzed and an approach for estimating the relative orientations and positions of the two cameras were given. Using such a process images can be readily rectified, before disparity calculation starts. In this chapter a number of possible algorithms for calculating binocular disparities will be evaluated. Emphasis is put on both speed and accuracy, knowing that the final system is intended to work in a dynamic environment under real-time conditions. Dense disparity maps are important, but an increased number of false positives, should not be traded for maximum density. If information about a certain image region is sparse and disparities cannot be reliably estimated, it is better to discard these results. Additional cues and estimates from different instances in time, will hopefully aid in handling situations when disparity calculation fail. In these regards this study is somewhat different from other studies. In the next section a historical overview will be given about computational studies dealing with binocular stereo. A number of methods based on area correlation will be tested in Section 5.2, with results compared to a method based on dynamic programming (Section 5.3) and a recently presented cooperative approach in Section 5.4. The chapter ends with final remarks and suggestions for further research.

5.1 Related work

The reason why people have studied binocular disparities as a cue to depth perception, has varied during the years. In earlier computational studies, scientists were interested in computational models of the human vision system and not so much in applying the results practically in reconstruction, surveillance or tracking human motion. One likely reason for that could be the lack of computational power typically required for such applications. Despite these limitations, binocular disparities was used for rover navigation and obstacle avoidance as early as in the late 70's (Hannah 1974, Gennery 1980). Moravec (1977) used feature points extracted as local maxima of a cleverly designed directional variance measure. The matching was done in a coarse to fine fashion using cross correlations. Even if the rover was slow, they were still able to show that disparities as means to control the motion of a rover is indeed feasible.

Inspired by the work of Julesz (1971), Marr & Poggio (1976) presented a cooperative model of the human stereo perception. Using a network of inhibitory connections between neurons representing the same image position, but different disparity. A uniqueness constraint was imposed, such that each image point could only be assigned a single disparity. Another constraint based on the assumption that nearby image points typically originate from nearby points in 3D space, was imposed through excitatory connections between neighbouring neurons of similar disparity. This constraint is known as the continuity assumption and is similar to the smoothness constraint in optical flow studies. A modern version, given in the cooperative model by Zitnick & Kanade (2000), shows very promising results.

In another study Marr & Poggio (1979), instead of using every image pixel, matched edge pixels only. This was done in a single pass, rather than in an iterative procedure such as in the cooperative model. The left and right images were filtered with twelve different orientation sensitive difference of Gaussians filters at various scales. Matching was performed using a similarity measure based on filter signs and orientations, with image positions defined by zero-crossings in the filtered images. Grimson (1981) implemented this model on faster hardware, but used a single Laplacian of Gaussian filter and grouped the edge pixels based on the orientation of the filtered responses.

The model was later extended (Grimson 1985), such that instead of imposing the continuity constraint over an area, it was done only along contours, as suggested by Mayhew & Frisby (1981). Prazdny (1985) presented an algorithm based on a more general constraint, called the coherence principle. This principle recognizes that nearby points may in fact come from different objects in 3D space. Thus it includes no inhibition between nearby points that do not agree on the same disparity, even if points of similar disparity still cooperate. The same is true in the PMF algorithm by Pollard et al. (1985), who in addition used the disparity gradient limit (Burt & Julesz 1980).

With the introduction of dynamic programming methods, dense disparity maps became more common. In order to better handle occlusions, Baker & Binford (1981) divided each edge into its two sides and considered the sides separately. These so called half-edges were matched between the left and right images using the Viterbi algorithm and then the gaps between edges were filled in using a second Viterbi pass. With the assumption that the ordering of edges along an epipolar line does not change, which is known as the ordering constraint, the matching complexity could be significantly reduced. In cases of very thin objects located in the foreground, this constraint does not always hold, but such occasions are rare. Ohta & Kanade (1985) used dynamic programming to match scanline intervals based on pixel brightness. These intervals were delimited by edge points and the consistency between edge points on different scanlines was enforced using a second dynamic programming process.

Through a Bayesian treatment of the matching problem along a scanline, it is possible to better deal with occlusions (Geiger et al. 1992, Belhumeur & Mumford 1992). These methods used shiftable windows to impose continuity in disparities, which is unlike the method of Cox et al. (1992), where only pixel intensities were used in conjunction with a second constraint that minimizes discontinuities between scanlines. A common problem with dynamic programming is the fact that a single mismatch at one point affects the following points of the same scanline, which results in the typical streaking effect of such methods. This is especially true in the case of very large disparities. In order to cope with this problem Intille & Bobick (1994) introduced the notion of highly reliable ground control points, that are used to guide the matching process.

Area-based methods have played a major role in practical applications during the years. Typically small areas are correlated between the left and right images, and matches are chosen based on maximum correlation. The benefit of such an approach is the simplicity, that makes implementation on fast hardware easier. Some of the first such systems were built at SRI (Quam 1984, Hannah 1985). While these systems used cross-correlations, Nishihara (1984) considered only the sign of Laplacians and got reasonable results. He also showed that the probability of mismatches decreases as the correlation window increases in width, but at the cost of lower accuracy. To overcome this problem it is possible to perform the matching in both directions, left to right and then in the opposite direction (Cochran & Medioni 1992, Fua 1993). Disparities are then accepted only for those pixels, where the matchings in both directions agree. A real-time system from INRIA (Faugeras et al. 1993) was based on this principle.

Instead of using only two cameras one may apply a whole series of cameras, arranged either on a row (Okutomi & Kanade 1991) or in an array (Satoh & Ohta 1996), without being forced to sacrifice accuracy for occlusion detection. In a binocular setting a wider baseline is typically required to achieve high accuracy, but the matching is then seriously complicated in cases of occlusions. There has been other attempts to deal with this problem. Zabih & Woodfill (1994) introduced what they called the rank and census transforms, which are more robust to outliers. Areas are matched based on the relative ordering of pixel intensities, rather than on the intensities themselves. Birchfield & Tomasi (1998) used a dissimilarity measure which is less sensitive to image sampling. This is done in order to cope with the fact that correlations are performed in steps of integer disparities, which might result in large errors in highly textured regions.

Just like in the case of optical flow (see Chapter 2), local phase information might be used for disparity calculation (Sanger 1988, Fleet et al. 1991, Maki et al. 1995). The benefit of using such information is its stability with respect to affine deformations and contrast change, as well as subpixel accuracy. However, the range of possible disparities is limited to the size of the filter kernels used. Thus phase information could be an interesting alternative if disparities are known to be small and high accuracy is required. This is, however, not the case in typical indoor environments. More general multiple linear filters for disparity calculation were proposed by Jones & Malik (1992).

During the last few years, more efforts have been devoted to the disparity calculation problem globally. Thanks to dynamic programming it was possible to optimize pixel matches along scanlines. However, this method is unfortunately limited to one-dimensional problems and cannot be applied over the whole image. An alternative global method was presented by Barnard (1989), who solved the problem using simulated annealing. However, even in a coarse to fine framework, this method is very slow.

With the introduction of graph cut based methods (Roy & Cox 1998, Ishikawa & Geiger 1998), greater speed could be achieved. In these studies a 3D network of nodes was created, with links between nodes representing the cost of introducing occlusions and differences in pixel intensities. The problem was formulated as letting a 2D plane divide the network into two halves, such that the total cost of links cut is minimized. Unfortunately, the cost of a discontinuity increases linearly as a function of the difference in disparities between two neighbouring pixels. Boykov et al. (2001) instead formulated the problem as a large labeling problem, with a greater flexibility in terms of possible cost functions. The problem was iteratively solved as a maximum flow problem, with one minimum cut determined at each iteration. Several recent and successful methods are of this type. Unfortunately, they are still too slow for real-time execution.

More information on the performance of the most popular methods for dense disparities can be found in a recent study by Scharstein & Szeliski (2002), which is a continuation of the comparative work in (Szeliski & Zabih 1999). A taxonomy is presented, together with numerous quantitative results of five separate methods, that represent different levels of algorithmic complexity. A prior study by Dhond & Aggarwal (1989) gives a thorough presentation of the historical background. It includes few quantitative results, unlike (Bolles et al. 1993) where three real-time methods from SRI, INRIA and Teleos were analyzed and tested for the application of planetary rover navigation.

5.2 Area-based methods

There are several reasons why methods based on spatial correlation by design give limited accuracy. First of all, since correlations are done pixel-wise, they tend to be biased towards integer disparity values. This problem might be overcome by fitting a curve, typically of second degree, to the correlation data and find a maximum along the curve. However, it might be difficult to find a theoretical motivation for the choice of a certain curve, even it the accuracy indeed is improved. In fact, the possible accuracy is determined by the Nyquist sampling limit, which affects any area based method. Another reason why curve fitting is not always enough is that image noise is usually not uncorrelated between different pixels, which leads to a bias towards the integer position closest to the true disparity value.

A second problem in correlation, is the fact that an aggregation area is required. The area must be large enough for the signal to noise ratio to be low. On the other hard, if the area is too large the probability will increase for pixels of different disparities to be covered by the same aggregation area, resulting in a competition between different disparity estimates. This is especially the case near occlusions. A larger area may also result in foreground objects being enlarged. Since a discontinuity edge belongs to a foreground object, nearby pixels in the background will be influenced by the erroneous disparity to the foreground object. The enlargement will increase if the background lacks texture. Thus a correlation based approach would benefit from the use of adaptive area sizes (Kanade & Okutomi 1994, Scharstein & Szeliski 1996).

In conclusion, there is a number of reasons for not choosing a method based on aggregation areas. However, area correlation has one major advantage, namely its simplicity, which may result in high speed. As mentioned in the chapter about optical flow a fast solution is sometimes more important than high accuracy. The solution should in the end be judged by the performance of the complete system, not of each individual component. If computations can be performed at a higher frame-rate, we typically get less complex tasks to solve, as the system needs to adapt to changes in the environment. This implies that even less accurate methods may prove to be more successful. Hence such approaches need to be considered.

Matching criteria

Before implementing an area-based disparity system, it is important to select a proper matching criterion. There are several alternatives to choose from. When evaluating the possibilities in terms of both accuracy and computational cost, it is not obvious what criterion is the overall winner. The most commonly used criteria are the sum of squared differences (Anandan 1989) and the sum of absolute differences (Kanade & Okutomi 1994, Konolige 1997), which can be expressed as follows:

$$C_{SAD}(\mathbf{p}, d) = \sum_{(x,y)\in W_\mathbf{p}} |I_l(x,y) - I_r(x+d,y)| \quad \text{and} \tag{5.1}$$

$$C_{SSD}(\mathbf{p}, d) = \sum_{(x,y)\in W_\mathbf{p}} (I_l(x,y) - I_r(x+d,y))^2, \tag{5.2}$$

where $I_l(x,y)$ and $I_r(x,y)$ represent the left and right images and $W_\mathbf{p}$ is a left image aggregation window around a point $\mathbf{p}$. Another matching criterion, with the characteristic feature of being invariant to image brightness variations, is normalized cross correlation (Hannah 1985, Faugeras et al. 1993)

$$C_{NCC}(\mathbf{p}, d) = \sum_{(x,y)\in W_\mathbf{p}} \frac{I_l(x,y)\, I_r(x+d,y) - \overline{I_l}(x,y)\, \overline{I_r}(x+d,y)}{\sqrt{\sigma_l^2(x,y)\, \sigma_r^2(x+d,y)}}. \tag{5.3}$$

Here $\sigma_l^2(x,y)$ and $\sigma_r^2(x,y)$ denote the variances of pixel intensities of the left and right aggregation windows. Since the two images come from different cameras, that might be set up somewhat differently, invariance to brightness change is often desirable. However, invariance may prove to be a too strong condition and result in windows of very different brightness being matched. A better alternative could thus be modified normalized cross-correlation (Moravec 1981),

$$C_{MNCC}(\mathbf{p}, d) = 2 \sum_{(x,y)\in W_\mathbf{p}} \frac{I_l(x,y)\, I_r(x+d,y) - \overline{I_l}(x,y)\, \overline{I_r}(x+d,y)}{\sigma_l^2(x,y) + \sigma_r^2(x+d,y)}, \tag{5.4}$$

that differs from $C_{NCC}(\mathbf{p}, d)$ by a factor of

$$2\frac{\sqrt{\sigma_l^2(x,y)\, \sigma_r^2(x+d,y)}}{\sigma_l^2(x,y) + \sigma_r^2(x+d,y)}, \tag{5.5}$$

that is the quotient of the geometric and arithmetic averages. Thus too large a difference between variances will be penalized in the matching process.

A final criterion, at least in this short exposé of criteria, is one due to Birchfield & Tomasi (1998), that tries to adjust for the fact that windows are only correlated in steps of integer disparity values. For the previously mentioned criteria, the difference between correlated windows may still by significant in highly textured regions. In this approach a range of acceptable intensities is calculated for each pixel of the right image and a match is evaluated based on how much a left image pixel departs from this range. First the midpoints between a right image pixel and its neighbours along a scanline are found, that is $I_r^-(x,y) = \frac{1}{2}(I_r(x-1,y) + I_r(x,y))$ and $I_r^+(x,y) = \frac{1}{2}(I_r(x,y) + I_r(x+1,y))$. The range is then defined as the image intensities between $I_r^\downarrow = \min(I_r^-, I_r^+, I_r)$ and $I_r^\uparrow = \max(I_r^-, I_r^+, I_r)$. The final criterion looks as follows:

$$C_{BT}(\mathbf{p}, d) = \sum_{(x,y)\in W_\mathbf{p}} \max(0, I_r^\downarrow(x,y) - I_l(x+d,y), I_l(x+d,y) - I_r^\uparrow(x,y)). \tag{5.6}$$

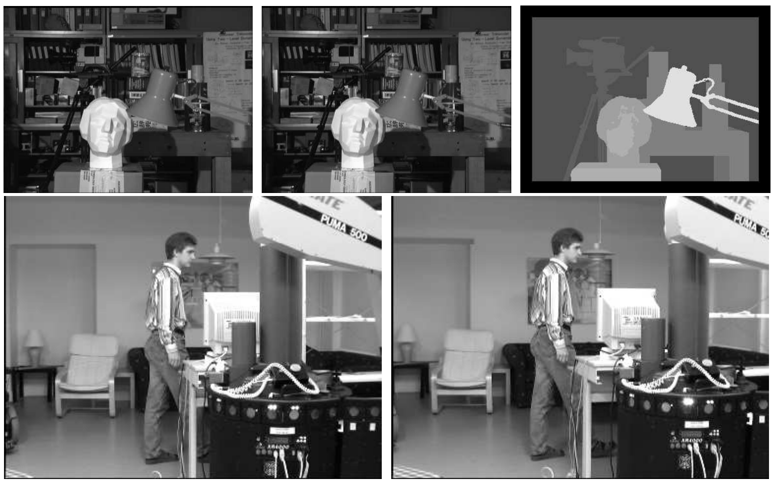

Figure 5.1. Two stereo pairs on which experiments are performed, the upper row showing the Tsukuba pair including ground-truth data and the lower from a typical stereo sequence.

Experiments

The above mentioned criteria, except normalized cross-correlations, have been tested on two different stereo pairs. The first one, originating from the University of Tsukuba, is given with a ground-truth disparity map. The original pair is in colour, but in this study a black and white version will be used instead, in order to keep the complexity of chosen methods to a minimum. The other pair comes from a typical indoor sequence and is intended to represent a more likely working condition. With a baseline of about 12 cms, the maximum disparity is as large as 52 pixels, which is considerably more than in normal test cases. Even if there are exceptions (Intille & Bobick 1994, Sara 1999), benchmarks are typically performed on stereo pairs similar to that of the Tsukuba one, with disparity ranges in the neighbourhood of about 16 pixels.

The two pairs, including the ground-truth data, are shown in Figure 5.1. In all experiments 7×7 pixels image windows were used. The size was chosen such that the calculated disparities were of acceptable quality, without any unnecessary smoothing of the resulting data. Thus one might say that high density is sacrificed for a better localization of discontinuities, with the hope that disparities may be easier segmented in objects located at different distances from the observer. Subpixel disparity estimates are found fitting the data to a second degree curve around the maximum. Results are shown without postprocessing in order to better illustrate the true performance of the matching process.

Resulting disparities can be seen in Figures 5.3 and 5.4, for runs using C_{SAD}, C_{SSD}, C_{MNCC} and C_{BT}, in the order from upper-left to lower-right. The first striking observation is that the results are almost identical, which is also con-

	C_{SAD}	C_{SSD}	C_{MNCC}	C_{BT}
Correct	68.3%	69.2%	69.2%	69.0%
Empty	23.4%	22.4%	22.8%	23.1%
Wrong	8.3%	8.4%	8.0%	7.9%

Figure 5.2. Number of correct, empty and incorrect estimates for different matching criteria on the Tsukuba stereo pair.

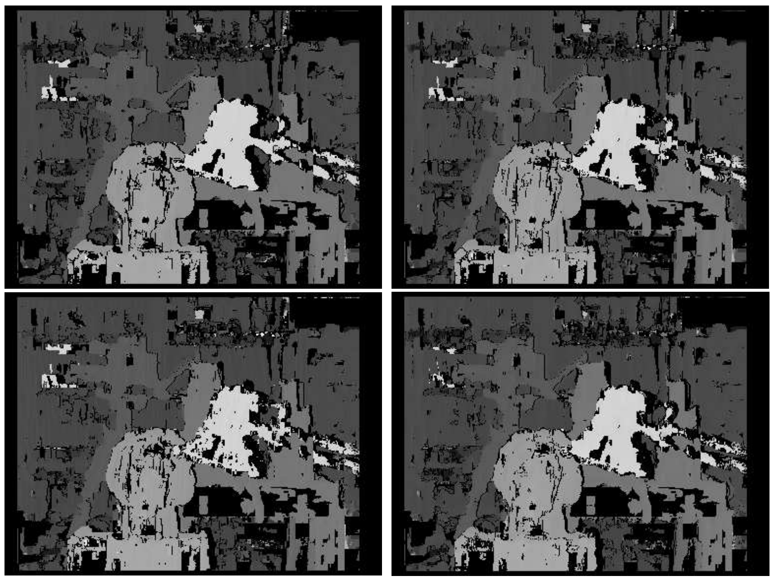

Figure 5.3. Results from the Tsukuba pair, using the matching criteria C_{SAD} (upper-left), C_{SSD} (upper-right), C_{MNCC} (lower-left) and C_{BT} (lower-right).

firmed by the table in Figure 5.2. An estimate is said to be correct if the difference compared to the ground truth is less than or equal to one pixel. Pixels are assigned as being empty, if either the variance of pixel intensities within the corresponding window is too small, or if they fail a left-to-right test. That is, correlations are performed from left to right and then back from right to left, and if the winners in each direction do not agree, the disparity estimate is considered as unreliable. For both sets of images, the method based on C_{BT} seems to result in cleanest disparity maps, but it is questionable if the increase in complexity is justified by the relatively small improvement in accuracy. From the results in Figure 5.4 it looks as if the C_{MNCC} criterion has more problems with reflexions on the floor, compared to the other three criteria. The results of C_{SAD} and C_{SSD} are much similar, except that slightly more pixels fail the left-to-right test using C_{SAD}.

In conclusion, there is no obvious choice in either case. The differences are so small that other test cases might have resulted in a totally different ranking

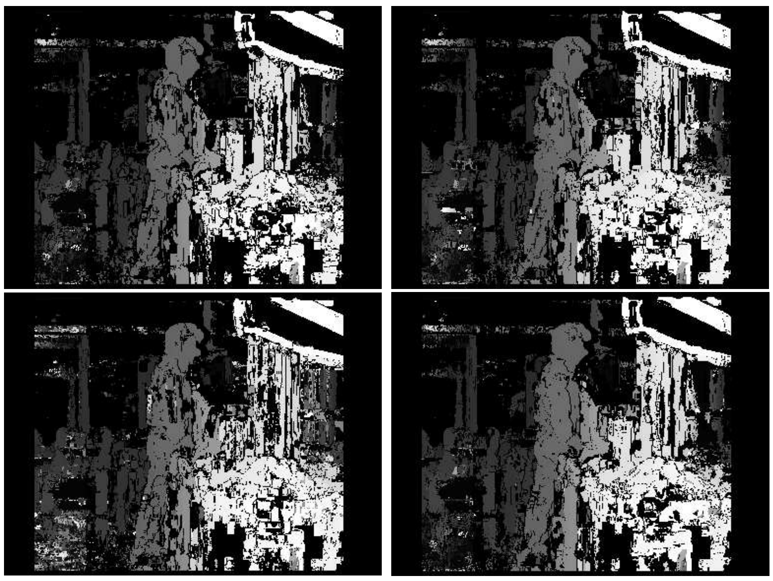

Figure 5.4. Results from a typical sequence, using the matching criteria C_{SAD} (upper-left), C_{SSD} (upper-right), C_{MNCC} (lower-left) and C_{BT} (lower-right).

of the criteria. The question that remains is if the methods can be separated based on computational costs. Clearly C_{SAD} and C_{SSD} are far less complex than the other two criteria. Which one to choose depends on the hardware on which the disparity estimation is supposed to be implemented, that is if one subtraction followed by taking the absolute value is faster than one multiplication. On number-crunching processors such as DEC Alpha 21264, C_{SSD} is likely to be the best choice. However, if the implementation is on an AMD Athlon or Intel Pentium 4 using the MMX instruction set, C_{SAD} is probably a better alternative, since MMX includes operations suitable for multiple absolute differences in parallel.

Window sizes

So far all experiments have been performed using 7×7 pixel aggregation windows. The windows were chosen large enough for image noise to be suppressed, but still limited in size not to complicate the separation of objects into foreground and background objects. In order to justify such a small aggregation window, a number of experiments using somewhat larger windows were performed. For example, in the study of Scharstein & Szeliski (2002), windows as large as 21×21 pixels were used, but it is unclear what motivated such a choice. Possibly it was in order to increase the density of data, so as to make comparison with other dense disparity method easier.

	7×7	11×11	15×15	19×19
Correct	68.3%	74.9%	77.8%	77.6%
Empty	23.4%	17.0%	13.8%	13.5%
Wrong	8.3%	8.0%	8.4%	8.9%

Figure 5.5. Number of correct, empty and incorrect estimates using different sizes of aggregation windows on the Tsukuba stereo pair.

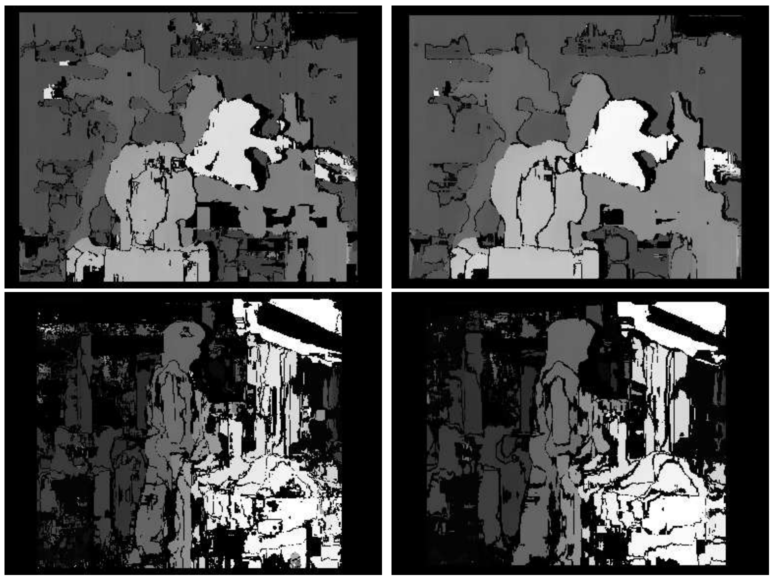

Figure 5.6. Disparity maps calculated using aggregation windows of dimensions 11×11 (left) and 15×15 pixels (right) for two different stereo pairs.

The table of Figure 5.5 shows a number of quantitative results based on the Tsukuba stereo pair. The major difference between sizes seems to be in the number of correct and empty estimates, since the incorrect ones only vary slightly. For this particular stereo pair the minor change in erroneous disparities is understandable, since image texture is usually located on both sides of existing discontinuities. Thus background and foreground compete and results are not spread beyond boundaries, enlarging objects in the foreground. As seen in Figure 5.6, this is not really the case for the second stereo pair. The head of the walking person is enlarged, as the background is lacking texture to counteract the dominance of the foreground. This would possibly have resulted in an increase in errors, if ground truth data were in fact available. In both cases discontinuities seem to get rounded as the size increases, which might be acceptable but also a problem depending on the application.

In order to understand the negative aspects of an enlargement, one might consider the following scenario. Assume that disparities are to be used in conjunction with another cue, such as colour, and both cues are combined for foreground-background segregation. Since colour data have highest confidence within areas of uniform colour and disparities are more accurate along edges, it might to valuable to combine these two cues for better segregation. Statistics could then be collected from foreground and background, and segregation be performed based on this information. This would typically result in a problem of separating different peaks in a colour-disparity space. However, if objects are enlarged these peaks may be less distinct and the foreground harder to identify. Thus sparse data kept within discontinuities may be more favourable in practice, even if denser data typically are required.

Postprocessing

In order to better preserve disparity discontinuities and make estimates more tolerant to image noise, so called shiftable windows have been used in a number of studies (Kanade & Okutomi 1994, Intille & Bobick 1994). During the earlier experiments is was assumed that an aggregation window is defined such that the point taken into consideration is located at the centre. this is not always the case. If the size of a window is too large, data across discontinuities risk being smoothed out, resulting in less reliable estimates. That occurs because pixels belonging to the background and foreground objects are present within the same aggregation window. If the centre of a window is instead allowed to move, it is possible to place it such that all pixels belong to either side.

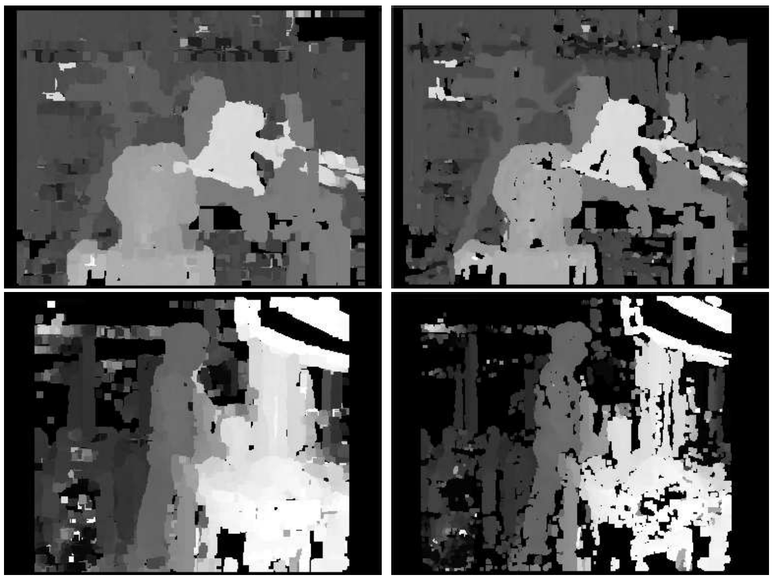

Figure 5.7. Results when postprocessing of calculated disparities is performed using shifted windows (left) and a local median filter (right).

One way of approximating such an approach, is performing a postprocessing with a two-dimensional max-filter on the correlations corresponding to the calculated disparities. If the size is the same as that of the aggregation window, the result is equivalent to the process of letting the centre move across the whole window and choosing the disparity that results in the highest correlation between windows in the two images. The final disparity estimate is then the result of the maximum correlation among all possible disparities and choices of window centres. However, since the same correlation will be used for several pixels, a distinct peak in correlation space may affect the results of all these pixels. This may potentially result in considerable errors, as can be seen in the left column of Figure 5.7. In areas of low contrast a single erroneous estimate may dominate its neighbourhood causing the error to spread. That is especially true near the ceiling in the upper row of the figure, where small erroneous blocks are clearly visible. The results are much more attractive in areas rich on image texture.

Another possibility is letting a median filter run over the whole disparity map. In the previous case the max-filter was working on the correlation data, which means that such data have to be kept until the postprocessing is finished. If instead a median filter is used on the resulting disparities, correlation data do not have to be stored. This means that less temporary data will be needed. The results in the left right column of Figure 5.7 come from such a procedure. The disparities are not as dense and the errors are still enhanced, but based on the table in Figure 5.8, they are at least fewer than with a shiftable window. For a median filter of size 5×5 the fraction of correct data is in fact higher than in case of the min-filter, even if the error-rate is lower. The filter was implemented such that in reality the median of medians are calculated, first along the x-axis and then the y-axis. This was done in order to keep the computational cost to a minimum, since calculating the median of 25 different values for each pixel can indeed prove costly. Since empty data are disregarded by the median filter, a slight blockiness is visible around areas of low contrast.

	Without	Shifted	Median 3	Median 5
Correct	68.3%	79.8%	78.4%	81.8%
Empty	23.4%	3.8%	10.2%	5.7%
Wrong	8.3%	16.4%	11.4%	12.5%

Figure 5.8. Number of correct, empty and incorrect estimates using different sizes of aggregation windows on the Tsukuba stereo pair.

5.3 Dynamic programming

In Section 5.2 disparities were only calculated based on local information, which meant that no reliable estimates could be found in image regions without enough texture. A better solution would instead be a search performed globally, finding the most likely combination of disparities over the whole image. This typically leads to iterative methods that unfortunately are too slow for a real-time requirement. Thus one might instead aim for an approach that optimizes scanline by scanline.

A commonly used assumption is the so called ordering constraint, which requires the ordering of scene points projected onto the two image planes to be the same. This means that if a point $\mathbf{x}_1$ is projected to the left of $\mathbf{x}_2$ in the left image, then it should be to the left in the right image as well. As mentioned earlier this does not always hold. If variations in depths are large and the scene includes thin objects in the foreground, the opposite situation might occur. This can easily be seen if two fingers are put in front of your eyes, one finger far behind the other. Both fingers are visible, but differently ordered in the eyes. Fortunately, these occasions are few in a real situation. If the constraint is supposed to hold, one might use it to speed up the process.

The squares in Figure 5.9 each represents a hypothesis of a pixel at position x_l in the left image being matched to x_r in the right. If they are not matched, there is an occlusion in either of the two cameras. Thus there are three different hypotheses for each square; match, left occlusion and right occlusion. An occlusion in one image corresponds to a discontinuity in disparity in the other. The two graphs illustrate the same thing, but with different choices of y-axes. The left and right graphs use right image positions and disparities respectively. If the ordering constraint is satisfied, one may conclude that if a pixel match hypothesis is true, then it is only possible for hypotheses in the upper-left and lower-right quadrants to hold, as seen in the left graph. Thus it is possible to order the hypotheses, such that hypotheses to the upper left precede those in the lower right quadrant. Such an ordering is what is required for a dynamic programming method to be used.

With the uniqueness and ordering constraints taken into consideration, a total matching of pixels along a scanline can be illustrated by a path from left to right in the left graph of Figure 5.9. Diagonal segments represent matches between the left and right cameras, that is the corresponding points are visible from both cameras. A horizontal line indicates an occlusion in the left camera, while a discontinuity results in a vertical line. Thus the problem is reduced to that of finding the lowest cost path from left to right. A cost function may be derived from maximum likelihoods (Cox et al. 1996), but in this study we instead express the function directly as

$$C(\mathbf{d}) = N_o \kappa_o + \sum_{i=1}^{N_m} (I_l(x_i) - I_r(x_i + d_i))^2. \tag{5.7}$$

The total cost is expressed as a function of disparities $\mathbf{d} = (d_1, d_2, ..., d_N)^\top$, where N is the number of pixels along a scanline. The parameter N_o is the number of left and right occlusions, that is a sum of vertical and horizontal segments in the path, and κ_o is the cost of each such occlusion. The number of matched pairs is N_m and squared differences of pixel brightnesses are used as the cost of a match. Due to the ordering of hypotheses, the cost at a particular

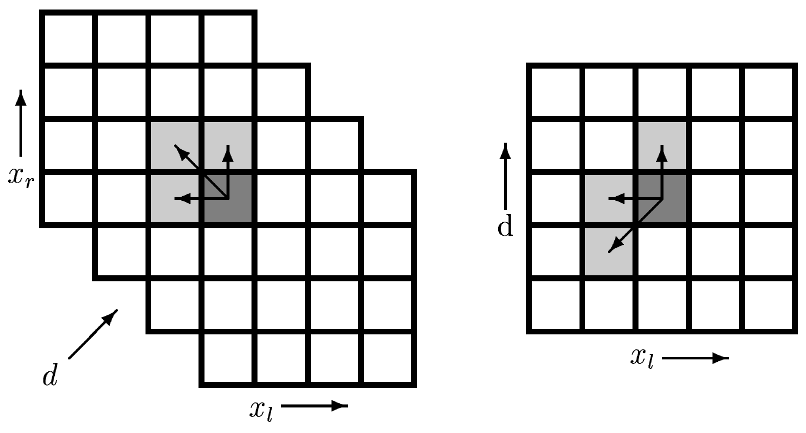

Figure 5.9. Squares representing combinations of left and right image pixel matches. The left graph uses right image positions as y-axis, while the right uses disparities. The arrows indicate local hypotheses of matching pair (diagonal), occlusion (horizontal) and discontinuities (vertical), as seen in the left graph.

location depends only on hypotheses in the directions of the arrows shown in the figure. If $C_{i,j}$ denotes the total accumulated minimum cost from the left to a point (i, j), an expression can be given as a function of the immediately preceding neighbours,

$$C_{i,j} = \min(C_{i-1,j-1} + (I_l(x_i) - I_r(x_j))^2,\ C_{i,j-1} + \kappa_o,\ C_{i-1,j} + \kappa_o). \quad (5.8)$$

The approach is typically implemented such that each square is visited from left to right and the minimum accumulated cost and winning hypothesis is temporarily stored, assuming that the path will pass the square. When the process ends, there are a number of possible paths, each ending at squares corresponding to different disparities of image pixel x_N. The path of lowest cost can thus be found and reconstructed following the segments back from right to left. The two images in Figure 5.10 show the estimated disparities using these methods on the two examples in Figure 5.1. The occlusion cost κ_o was chosen such that the best overall performance was achieved using the same cost for both stereo pairs. For the two dotted scanlines in the right image of Figure 5.10, the corresponding matching costs are shown in Figure 5.11, that is the squared difference of matched image pixels. Here the vertical axis represents different disparities, as in the right graph of Figure 5.9. The minimum cost paths are shown in white with diagonal and vertical gaps indicating the existence of occlusions and discontinuities.

One of the drawbacks with the approach just described is the fact that disparities are estimated scanline by scanline without any coupling between results from different scanlines. This often results in a streaking effect, when different

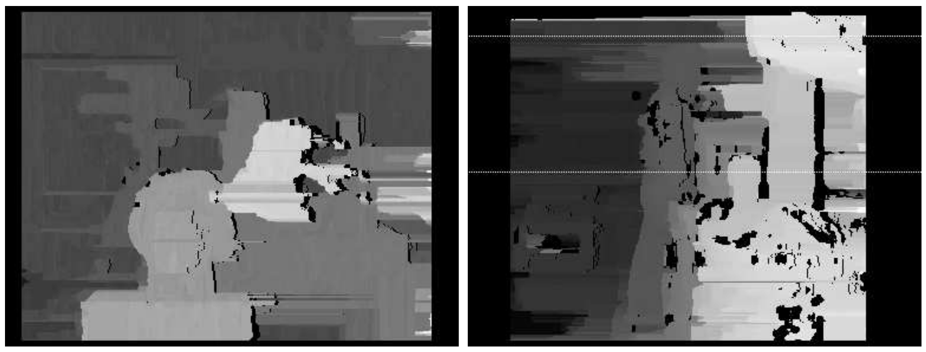
Figure 5.10. Calculated disparities from the two stereo pairs in Figure 5.1 using dynamic programming. The dotted lines show scanlines for which matching costs are shown in Figure 5.11.

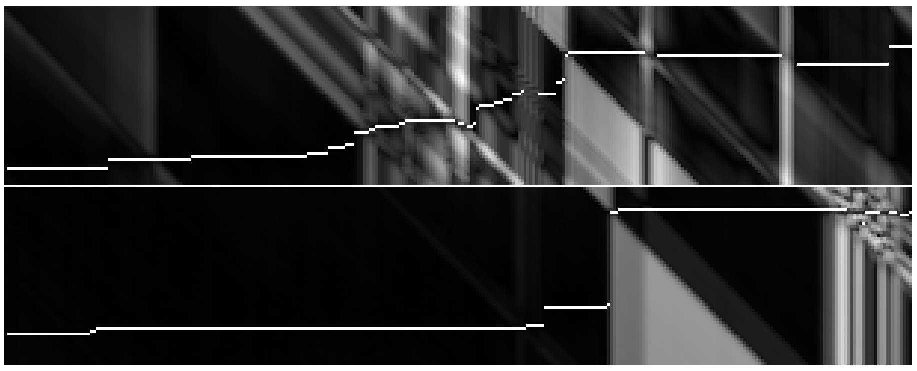
Figure 5.11. The squared differences between matched image pixels of the left and right cameras, for the two scanlines in Figure 5.10. The vertical axes represents different disparities and minimum cost paths are shown in white, as in the right graph of Figure 5.9.

competing matchings of similar cost win for different scanlines. In the results of Figure 5.10, the matching costs are thus calculated such that summations are performed on squared differences in 7×7 pixel windows, similar to what is described in Section 5.2. Even if streaking is significantly reduced, it is still evident. Larger windows typically do not improve the results much and may also lead to occlusions being harder to separate from disparity gradients of curves surfaces. Considerable errors are visible to the right of the autonomous platform in the right image of Figure 5.10. There is not enough texture in the background to compensate for the cost related to the large differences in disparity between foreground and background. The quantitative results of the Tsukuba pair indicate similar problems. Only 2.8% of the estimates are empty, 82.5% are correctly matched and the remaining 14.7% are thus incorrect, which is a far greater error-rate than for the area-based methods.

Dynamic programming methods have often been considered attractive, since they lead to dense disparities and still allow occlusion detection. However, they have one major flaw. A disparity calculated on a discontinuity typically originates from an object in the foreground. If there is no image structure in the background to support a hypothesis of a different disparity, the corresponding region will incorrectly assume the disparity of the foreground. This means that one might falsely believe that the foreground is larger that what is actually is. In area-based methods, such textureless regions would simply be disregarded. For these regions there is simply not enough information. With disparities optimized for over the whole image, support could have come from structure on different scanlines, which makes global optimization more robust.

If dynamic programming is still preferred to a more complex global method, it would be possible to use a quality measure based on local data. However, in the end this would lead to results much similar to that of area-based methods, since such a measure would typically be based on the local variance of either image pixels or correlation data. This means that uniformly shaded regions would be excluded and not much is gained.

5.4 Cooperative stereo

The last algorithm to be evaluated in this study is a cooperative method of Zitnick & Kanade (2000), which has a couple of interesting properties. The method is very simple in its implementation, while able to estimate disparities globally. Methods based on graph cuts are far more complex. On the other hand, the nonlinear nature of cooperative methods, makes their behaviours hard to predict. Unlike dynamic programming methods, the ordering constraint is not assumed, even if it can be incorporated in theory. Thus it has the potential of doing better on scenes with large variations in depths.

Cooperative method are built on the notion of excitation and inhibition regions in disparity-space, as illustrated in Figure 5.12. Nearby locations of similar disparity, shown in medium grey, cooperate in determining better disparity estimates, whereas different disparity assumptions for the same pixel position compete, which is shown in light grey. The inhibition regions $\Psi(x_l, y_l, d)$ are divided into two parts, one along the y-axis in the figure, corresponding to the same left image position, and a diagonal one for the same positions in the right image. For simplicity, regions are only shown along a scanline in Figure 5.12. The complete disparity-space is in fact three-dimensional, with an excitation region $\Phi(x_l, y_l, d)$ of dimension $5 \times 5 \times 3$.

The cooperative stereo algorithm shown in Figure 5.13 is iterative and maintains a set of matching scores during the whole process. The initial matching scores $S_0(x_l, y_l, d)$ are set using modified normalized cross-correlation between local windows in the left and right images. A threshold is used such that negative values are set to zero. Scores are sequentially summed over the excitation and

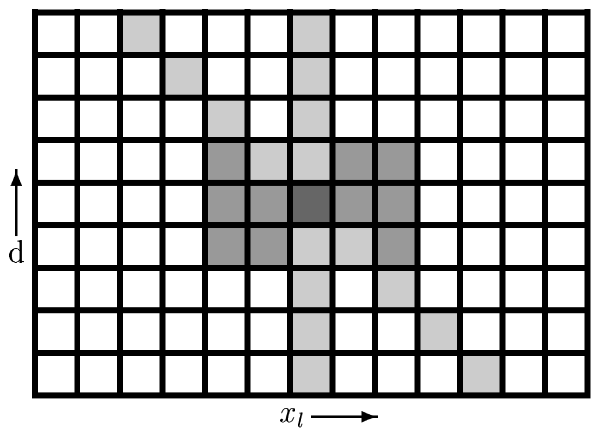

Figure 5.12. Disparity space image along a scanline, with inhibition $\Psi(x_l, y_l, d)$ and excitation $\Phi(x_l, y_l, d)$ regions shown in light and medium grey respectively.

inhibition regions, and finally updated. During the excitation phase scores are spread in the neighbourhood of each point. The inhibition in Equation 5.12 is performed such that normalization is first performed on all values in the inhibition region. The largest scores are then strengthened using an exponent α and finally the scores are weighted by the initial matching score.

Thus a score will never grow stronger than its initial value. The rational behind this is the following. A correct match will most likely have a high matching score, but an erroneous one does not necessarily lead to a low score. In areas of uniform texture, there might be numerous possible disparities that each result in high scores. Due to the competition and influence from nearby regions, one disparity value will end up being the winner, which is given by the disparity corresponding to the largest score. Before accepting an estimated disparity, its score is tested towards a predefined threshold. In regions of limited texture or occlusions, no single matching score is able to dominate other scores of the same inhibition region, which means that its maximum value will be too small. Consequently, these disparity values will be disregarded. Convergence is guaranteed using an exponent α greater than 1. With a value of 2 the algorithm converges in about 10 iterations. In fact, the choice of α does not significantly affect the accuracy of the final result, only the convergence rate.

Similar to the previously mentioned algorithm, the cooperative one was tested using the two stereo pairs in Figure 5.1 with results shown in Figure 5.14. For the Tsukuba pair for which ground truth is given, the fraction of correctly estimated disparities is 92.3%, 0.8% are empty and 6.9% erroneous. The errors are considerably larger than what Zitnick & Kanade (2000) reported in their study. Even if it is unclear from their report, the difference could be due to the fact that the original Tsukuba pair is in colour and here only the black and white version is used. The most challenging areas in this particular pair, assuming the

Cooperative stereo

1. Initialize matching strengths

$$S_0(x_l, y_l, d) = \max(S_{MNCC}(x_l, y_l, d),\ 0) \tag{5.9}$$

2. Sum over excitation regions

$$S_n^E(x_l, y_l, d) = \sum_{(x',y',d') \in \Phi(x_l,y_l,d)} S_n(x', y', d') \tag{5.10}$$

3. Sum over inhibition regions

$$S_n^I(x_l, y_l, d) = \sum_{(x',y',d') \in \Psi(x_l,y_l,d)} S_n^E(x', y', d') \tag{5.11}$$

4. Update matching strength

$$S_{n+1}(x_l, y_l, d) = S_0(x_l, y_l, d) \left(\frac{S_n^E(x_l, y_l, d)}{S_n^I(x_l, y_l, d)} \right)^{\alpha} \tag{5.12}$$

5. Until convergence, return to 2.
6. For each pixel (x_l, y_l) find d that maximizes $S_{n+1}(x_l, y_l, d)$

Figure 5.13. Cooperative stereo algorithm

occluding areas can be handled properly, are those of no texture. For example, the region just below the lamp contains no horizontal image gradients, except for the boundary of the shadow across the side of the table. Without this boundary it would hardly be possible to get an accurate disparity estimate. The information from the boundary is also spread to the region below the table, which is not correct, since this region belongs to the background. On both sides of the table information from the background has further been able to spread upwards. An improved result could probably be achieved if the excitation phase is only performed on regions of similar brightness or colour, but it is still very hard to understand how Zitnick & Kanade (2000) got their results, which indicate no such problems.

5.5 Conclusions

As could be seen from the experiments, the major difference between the evaluated methods is the density of the calculated disparities, rather than the accuracy itself. Another criterion that ought to be considered is the computational cost. Faster execution leads to computations at higher frame rate, which means that

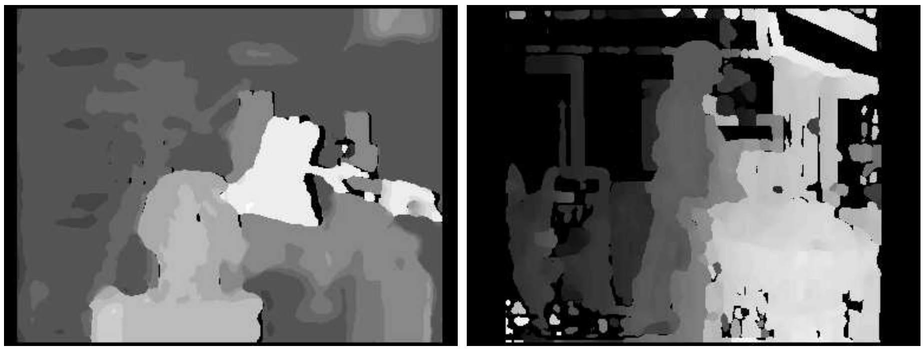

Figure 5.14. Calculated disparities from the two stereo pairs in Figure 5.1 using the cooperative method of Zitnick & Kanade.

	SAD	SSD	MNCC	BT	DP
Correlation	35 ms	45 ms	81 ms	175 ms	45 ms
Maximization	69 ms	69 ms	69 ms	69 ms	44 ms
Total	104 ms	114 ms	150 ms	244 ms	89 ms

Figure 5.15. Computational cost, on a 1.2 Gz Athlon, of five different methods, divided into the cost of area correlations and finding peaks in correlation space, with a maximum range of a 20 disparities on 384 × 288 pixel images.

possible temporal consistency may be exploited in the complete system. A summary of costs on an 1.2 GHz Athlon processor for the above mentioned methods, with an exception of the cooperative algorithm, is given in Figure 5.15. Data have been divided into the cost of performing area correlations and then finding the peak for which the correlation gives the maximum score. All methods were been implemented in C++, with correlation and brightness values represented as floating points. Optimization was only done such that the total allocation of memory was kept as low as possible, but no additional hardware specific optimization was done. A motivation for this decision can be found in the appendix.

Maybe surprisingly, the fastest method is the one based on dynamic programming. The reason is that no left-right test is performed with this method. Since the major portion of the Maximization cost is due to iterations over correlation values, which is done twice for the other methods, this phase is considerably faster in the dynamic programming approach. However, it is hard to determine any feasible quality measure to be used with this approach, in order to determine if estimates should be considered as trustworthy or not. Based on the qualitative results previously addressed, it is questionable if the additional costs of using modified normalized cross-correlation or the similarity measure of Birchfield & Tomasi (1998) is well motivated. The cost of the final approach, the cooperative one, is as high as about 2.6 s with 10 iterations. This is primarily due to the radical increase in allocated memory.

During the last decade there has been a surge for accurate dense disparity maps and not only data along edges or in other high contrast regions. This has primarily been motivated by the applications. However, for image regions of uniform brightness, these methods may at best compute the most likely disparity, given the support from nearby textured regions. Typically, constraints from continuity are imposed in order to fill in areas without such texture. This is often done through the use of global cost functions, where a certain positive cost is assigned to discontinuities in the resulting disparity map. Thus if two different hypotheses of similar costs are available based on brightness constraints only, the hypothesis that results in the fewest number of discontinuities will be chosen. This is true even if additional information, that might promote an introduction of discontinuities, is in fact available. Such information might originate from observing the optical flow field as the observer is moving about. The existence of T-junctions may also be considered as evidence for discontinuities and could in theory be incorporated into a minimization problem when disparities are estimated. Future methods should thus be opened up, so that such external information can be taken advantage of.

In conclusion, the major difference between the tested methods is the density and not the accuracy itself. Since our system does not require the density of the more complex methods, we used the simplest possible approach based on sums of absolute differences. A hardware specific implementation of such an approach has a computation cost of only 15 ms on an 1.2 GHz Athlon processor. Using the capabilities of the MMX instruction set, it was possible to parallelize most operations, with correlation data and image pixels stored as short integers. This method will be integrated into the complete system presented in Chapter 8.

Chapter 6

Ego-motion

In this chapter several well-known monocular structure-from-motion methods will be analyzed, in the context of an autonomous system moving around in an indoor environment. The intention is to determine whether such a method can be used for ego-motion estimation. One important reason why the system would benefit from knowing its ego-motion, is the ability to determine time-to-impact, that is the time required for the system to reach an observed object in 3D space. It will also be shown in Chapter 7 how the ego-motion can be used to detect regions of independent motion. A third reason is that ego-motion information may help the system localize itself in relation to a representation of the environment.

The two problems of determining 3D structure and estimating ego-motion are in fact interrelated. If one is solved, the other one follows relatively easily, at least in a static environment. The most evident coupling is the ambiguity in scale between translations and reconstructed depths. However, there are other less obvious ambiguities, especially in the case of noisy measurement data. As a consequence of these ambiguities, the performance of the evaluated methods will prove to be worse than what the intended application requires. It will also be shown that the difficulties are inherent to the problem itself, rather than the models and methods used to solve it.

Ego-motion in relation to epipolar geometry

The problems of estimating ego-motion and epipolar geometry, as discussed in Chapter 4, have much in common, in that they can be similarly formulated. Ego-motion can be determined from two consecutive images, whereas epipolar geometry is estimated from a stereo pair. There are, however, a number of important differences. Stereo involves a translation along the baseline, which is typically almost parallel to the image planes. In motion the translational direction is usually close to the optical axis. Depending on the translational direction, different algorithms may behave very differently under noisy conditions. This indicates that one should not necessarily use the same method for both problems. Another difference is that motion typically results in rather small image motion displacements, which make ego-motion estimation more sensitive to image noise. The ego-motion problem may be further complicated by independently moving objects present in the scene.

When estimating ego-motion, shorter translations make the problem harder (Weng et al. 1993, Oliensis 1996), whereas in stereo there will always be a translation along the baseline. This means that the ego-motion methods should gracefully adapt to cases of no motion. A natural way of relieving the difficulty of short translations, is by increasing the time between the considered image frames. However, this may lead to a more complex matching problem, since outliers due to independent motion, brightness change and occlusions become more likely. This is true for feature as well as dense optical flow based methods.

As described in Chapter 2, optical flow algorithms tend to collapse if displacements are too large. Furthermore, it can be shown that due to the aperture problem, the addition of more data using a smoothness constraint, does not significantly improve the quality of estimated ego-motion (Young & Chellappa 1992). This leads us to believe that a method based on feature points is preferable, since such a method would allow the computations to be performed at a higher frame-rate. In order to reduce noise sensitivity, a complete system could further include an extended Kalman filter on structure and ego-motion (Heel 1990) or on ego-motion alone (Soatto & Perona 1997). The work presented here primarily deals with the two-frame case, since recursive methods typically consist of multiple two-frame operations performed in sequence. However, it is acknowledged that a multi-frame batch approach is likely to be more successful for reconstruction of 3D geometry in an off-line environment.

6.1 The bilinear constraint

The epipolar constraint, that was discussed in Chapter 4, can in theory be used for monocular motion analysis as well. However, the constraint only works for wide enough baselines and it collapses if the motions are not sufficiently large. This has led some researchers to follow a different route and instead consider the optical flow field (Koenderink & van Doorn 1975) as a basis for determining motion and structure. If the flow field is differentiated (Longuet-Higgins & Prazdny 1980, Prazdny 1981), a bilinear constraint can be derived, relating optical flow vectors to motion and structure. Here we review such a derivation due to Bruss & Horn (1983).

If $\mathbf{X} = (X, Y, Z)^\top$ is a rigid point in 3D space, its corresponding perspective projection onto an image surface is given by $\mathbf{x} = (x, y)^\top = (X/Z, Y/Z)^\top$. The optical flow at $\mathbf{x}$ can then be expressed as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \frac{1}{Z^2} \begin{pmatrix} \dot{X}Z - X\dot{Z} \\ \dot{Y}Z - Y\dot{Z} \end{pmatrix}. \tag{6.1}$$

Assume the observer is moving in the scene with a translation along $\mathbf{t} = (t_x, t_y, t_z)^\top$ and rotating with an angular velocity $\omega = (\omega_x, \omega_y, \omega_z)^\top$. The velocity of $\mathbf{X}$, as seen by the observer, will then be given by $\dot{\mathbf{X}} = \mathbf{T} + \omega \times \mathbf{X}$ which can be written more explicitly as

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} t_x + \omega_y Z - \omega_z Y \\ t_y + \omega_z X - \omega_x Z \\ t_z + \omega_x Y - \omega_y X \end{pmatrix}. \tag{6.2}$$

A combination of Equations 6.1 and 6.2 yields

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} t_x - x\, t_z \\ t_y - y\, t_z \end{pmatrix} + \begin{pmatrix} -xy\, \omega_x + (1 + x^2)\, \omega_y - y\, \omega_z \\ -(1 + y^2)\, \omega_x + xy\, \omega_y + x\, \omega_z \end{pmatrix} \tag{6.3}$$

or, if written in a more compact form,

$$\mathbf{u} = \mathbf{u_t} + \mathbf{u_r} = Z^{-1} A(\mathbf{x})\, \mathbf{t} + B(\mathbf{x})\, \omega. \tag{6.4}$$

The translational and rotational components are related to the measurable image position of $\mathbf{x}$ by the corresponding two matrices

$$A(\mathbf{x}) = \begin{pmatrix} 1 & 0 & -x \\ 0 & 1 & -y \end{pmatrix} \quad \text{and} \tag{6.5}$$

$$B(\mathbf{x}) = \begin{pmatrix} -xy & 1 + x^2 & -y \\ -1 - y^2 & xy & x \end{pmatrix}. \tag{6.6}$$

As seen in Equation 6.4, with exception for the depth Z, the equation only contains information about $\mathbf{X}$ available from data available in the projection. In conclusion, the unknown parameters are the rotation ω, translation $\mathbf{t}$ and the depths of each individual image point. Unlike the essential matrix, structure has not been decoupled from motion, since the depths have not been eliminated. As seen in Figure 6.1, separating translation from rotation is far from trivial. A translation along the x-axis is similar to a rotation around the y-axis, even if the depths affect the magnitudes of the translational flow and the rotational flow is only approximately parallel to the x-axis. Because of this similarity, it is easy to understand that a translation parallel to the image plane is harder to determine than one along the optical axis.

Algorithms for estimating structure and motion can be divided into different categories. The optimization criteria can be either discrete or differential, linear or non-linear in the unknown parameters, with solutions that are either direct or iterative. In the forthcoming sections a number of methods will be evaluated, in order to judge whether the performance, in terms of speed and accuracy, is sufficient for real-time use. The following notation will be used in the description of methods. A true value of a measured parameter $\mathbf{x}$ is denoted $\underline{\mathbf{x}}$. A projection onto a surface defined by its normal $\mathbf{n}$ is written as a projection matrix $\mathbf{n}^\perp = \mathbf{I} - \mathbf{n}(\mathbf{n}^\top \mathbf{n})^{-1} \mathbf{n}^\top$. The matrix $\hat{\omega}$ is a skew-symmetric matrix, such that $\hat{\omega}\mathbf{x} = \omega \times \mathbf{x}$ for all vectors $\mathbf{x}$. A pseudo-inverse of a matrix $\mathbf{A}$ is denoted $\mathbf{A}^\dagger = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top$.

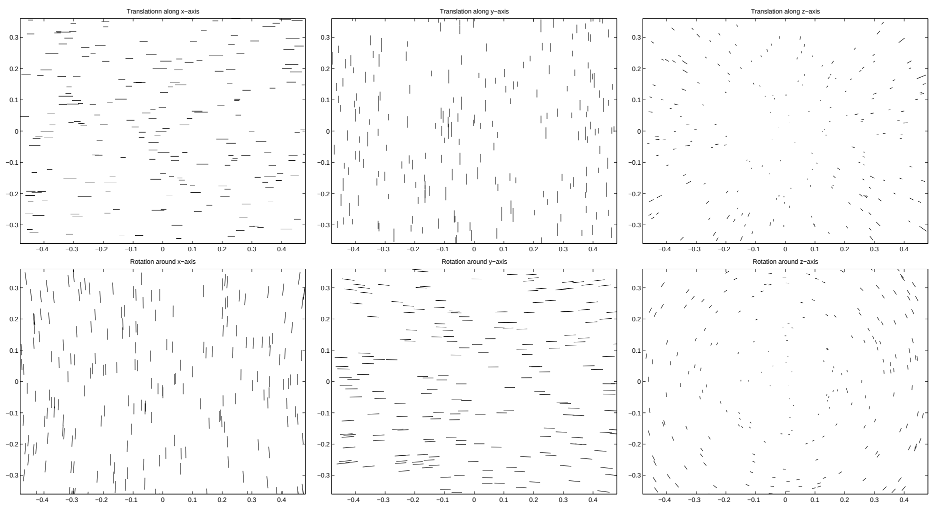

Figure 6.1. Typical flow fields for translations (upper) and rotations (lower) along and around the x-axis (left), y-axis (middle) and z-axis (right).

6.1.1 Instabilities

In a discussion such as this, ambiguities in the determination of structure and motion are usually treated either towards the end or during the presentation of results. In this study, however, the most important ambiguities are briefly discussed early on, in order to make the understanding of methods presented easier. These are in fact independent of the methods used and inherent to the problem (Adiv 1985*b*).

Bas-relief ambiguity

The most well-known ambiguity in structure-from-motion problems is the so called bas-relief ambiguity, which manifests itself in the difficulty in separating translations from rotations, for translations in the plane determined by the optical axis and the true translation (Jepson & Heeger 1990). For orthogonal projections the bas-relief ambiguity exists even in the noise-free case, which has been nicely illustrated by Belhumeur et al. (1997) and analyzed by Szeliski & Kang (1997) using the Hessian matrix.

First assume that there are no rotations or the rotations have already been adjusted for. Since depths can be chosen arbitrarily, an error in the flow will only be evident in the direction orthogonal to the true translational flow, which is determined by $(A(\mathbf{x})\underline{\mathbf{t}})^{\perp}$ (see Equation 6.4). An erroneous translation $\mathbf{t}$ will thus have a measurable error related to

$$N = (A(\mathbf{x})\underline{\mathbf{t}})^{\perp}(A(\mathbf{x})\mathbf{t}) = (x, y, 1)(\underline{\mathbf{t}} \times \mathbf{t}). \tag{6.7}$$

Even if N is only the nominator of the real error, since $(A(\mathbf{x})\underline{\mathbf{t}})^{\perp}$ have to be normalized, it captures the general behaviour of the error function for points not too close to the focus of expansion, where $A(\mathbf{x})\underline{\mathbf{t}} = 0$.

If the field of view is small, the terms of Equation 6.7 that include x and y will be small and the error is dominated by the last term, $\underline{t}_x t_y - \underline{t}_y t_x$. This means that a small translational perturbation in the plane determined by the optical axis and the focus of expansion, will result in a small measurable error. However, the location of the focus of expansion will change as a result of such a perturbation, which means that points near it may contain significant errors. Since the field of view is limited and the quadratic terms of the rotational matrix $B(\mathbf{x})$ are small, these errors may be compensated by a slight rotation around an axis orthogonal to $\mathbf{t}$ and $\underline{\mathbf{t}}$. The magnitude of the rotation is determined by the values of Z^{-1}. A compensation is only possible if all feature points are located at the same depth, which is typically not the case. However, if the depths are similar for all points, a compensation is possible and it will be harder to separate rotations from translations.

Flipped minima

Another ambiguity that has been observed in a number of studies, is the existence of double minima, one minimum on either side of the optical axis (Tomasi & Shi 1993). The most distinct minimum usually corresponds to the correct motion, whereas the second one is incorrect. This erroneous motion, often called the "rubber motion" percept, has been observed in psychological experiments. Soatto & Brockett (1998) characterize the behaviour as the inverted depths $d = Z^{-1}$ being mirrored around the mean inverted depth $\overline{d}$. With inverted depths being mirrored and the translational direction being on the opposite side of the optical axis, the corresponding optical flow takes the form

$$(2\overline{d} - d)A(\mathbf{x}) \begin{pmatrix} -t_x \\ -t_y \\ t_z \end{pmatrix} = d \begin{pmatrix} t_x \\ t_y \end{pmatrix} + (2\overline{d} - d) \begin{pmatrix} -xt_z \\ -yt_z \end{pmatrix} - 2\overline{d} \begin{pmatrix} t_x \\ t_y \end{pmatrix} \approx$$

$$d \begin{pmatrix} t_x - xt_z \\ t_y - yt_z \end{pmatrix} - 2\overline{d} \begin{pmatrix} t_x \\ t_y \end{pmatrix} \approx dA(\mathbf{x})\mathbf{t} + B(\mathbf{x}) \begin{pmatrix} 2\overline{d}\,t_y \\ -2\overline{d}\,t_x \\ 0 \end{pmatrix} \tag{6.8}$$

The approximations are due to the assumption that the field of view and depth variations are small. However, this does not hold if the translation is dominated by forward motion, which can also be seen experimentally. In the second row the mean translational flow has been compensated by an appropriate rotation. Thus there are two competing solutions that result in similar residuals.

There is, however, an error in the reasoning mentioned above. Even if the depth variation is significant, multiple minima can be seen experimentally. Oliensis (2000) analyzed the same problem using a series of approximations to describe the behaviour of the error function. One of his observations was that the error function tends to reach a maximum somewhere within the image. For points near the estimated focus of expansion, the denominator of the error function, $D = |A(\mathbf{x})\mathbf{t}|$, will be small and the real error $E = N/D$ large. If feature points are spread all over the field of view, which was assumed to be small, the combined influence on the error function of all features involved, will be a sharp peak close to the image centre. This was disregarded in the previous section.

However, due to the nominator N, it will still reach a minimum in the true translational direction and if this direction is close enough to the optical axis, the sharp peak will disappear. We further assumed that an error in translation can be compensated by a corresponding rotation, but this is only true for small errors in translational direction. The compensation errors will grow the further you go from the true direction, but the errors increase relatively slowly. The combination of a shallow error function due to N and a sharp peak close to the centre as a result of the denominator D, will result in two different minima on either side of optical axis.

In conclusion, the bas-relief ambiguity manifests itself in an instability in the translational direction within the plane defined by the true direction and the optical axis. Outside this plane, the errors are relatively large. An error in translation can be compensated by an appropriate rotation around an axis orthogonal to the optical axis and the true translational direction. The error will grow slowly as the necessary compensation increases. Since points are spread within a limited field of view and points near an erroneous focus of expansion contain significant errors, the error function will show a peak close to the centre of the image. As a result of these two factors, there will be a second minimum on the opposite side of optical axis.

6.1.2 Related work

During the years a number of two-frame methods based on the bilinear constraint have been presented. Some of these have been experimentally analyzed in e.g. (Tian et al. 1996). Bruss & Horn (1983) expressed the depths Z in terms of flow vectors, rotation and translation. They then established a least squares formulation with depths being eliminated and proposed a solution based on nonlinear optimization. This approach was later used by Adiv (1985*a*), who also segmented the scene into regions of different motions. Prazdny (1981) worked on the flow vectors directly, without explicitly considering the depths. He instead used an error measure based on the variance of pair-wise intersections of flow vectors, after being stabilized for different rotations. This approach is different from most other methods in that optimization is performed for rotations, rather than for translational directions.

A number of so called subspace methods have been presented by Heeger & Jepson (1992). These methods have been applied by a number of groups since they were introduced. What they have in common is that they try to cancel out one group of flow vectors, such that the result only contains information from the remaining ones. In (Heeger & Jepson 1992) a search is performed in the space of translations. For each translation, the corresponding rotation and depths are found directly. The optimization is then performed on the resulting errors. An alternative solution can be derived by finding a translation using the space orthogonal to that of all possible rotational flows (Jepson & Heeger 1992). A similar method using spherical projections, instead of projective ones, has been presented by Thomas et al. (1994).

Unfortunately, the combination of perspective projections and a cancellation of one group of flow vectors, typically results in a biased estimator. Even if spherical projections are unbiased in theory, they require the field of view to be as large as 180°. A number of attempts have been made to correct the bias in the perspective case. Jepson & Heeger (1992) suggest that dithering could be used, adding noise to the constraints making the total noise of the optimization function isotropic. However, adding additional noise to the already unstable problem feels intuitively unnatural and has therefore rarely been done in practice. Kanatani (1993*b*) instead used a rather complex iterative approach for subtracting the anisotropic covariance matrix prior to estimating the translation. Another method is based on pre-whitening, that is pre- and post-multiplying the measurement matrix with the square-root of its covariance matrix (MacLean 1999).

Separating rotation from translation can also be done by observing that the rotational flow is independent of depth. If each flow vector is expressed in terms of its corresponding angular velocity vector, the components due to rotation will be the same over the whole image. Prazdny (1983) used this observation and subtracted pair-wise velocity vectors, which led to constraints only containing information on the translation. Unfortunately, he never showed how this was to be done in practice. Rieger & Lawton (1985) used two nearby points on either side of an occlusion and subtracted their corresponding flow vectors. The difference is then only determined by the translation and point towards the focus of expansion. The problem with this approach is the difficulty of accurately estimating optical flow, especially near occlusions. Another method using pairs of image features is one of Tomasi & Shi (1993), who measured the change in angles between image features, since this change is also invariant to rotation. All these three methods have in common that they are based on analysis of motion parallax between pairs of image features. Unfortunately, such methods have shown to be especially sensitive to image noise.

Due to the inherent difficulty of separating rotations from translations, a method based on one component being estimated prior to the other will sometimes result in serious problems. This is simply because errors in the first component, result in errors in the next one and these errors will never be fully corrected. During the last couple of years a number of iterative methods have emerged that instead alternate between solving for rotation and translational direction. In such an approach it is possible to take the bas-relief ambiguity into account and jump from one minimum to another. Similar to Heeger & Jepson (1992), Oliensis & Genc (1999) found a rotation using the flow components orthogonal to a given translational direction. After compensating for this rotation a search is performed to update the translation and so on.

None of the previously mentioned methods keeps any information on depths during the optimization. Through series of algebraic transformations, the depths have efficiently been canceled out. This makes sense, since the depths can be chosen arbitrarily. However, there is a risk that the optimization function loses its relation to the image position noise. A notable exception from this rule is a method of Zhang & Tomasi (1999) who used the original error function, as defined by Equation 6.4, keeping the depths in the optimization function. This method instead alternates between estimating rotation, translation and depths. Another method that maintains the depth is a method based on spherical projections proposed by Chiuso et al. (2000). Like the method of Heeger & Jepson (1992) rotations and depths are found concurrently, using an estimated translation. However, the translation is then updated using not only the rotation, but also the depths. The benefit of these methods is the unbiasedness due to the choice of optimization criteria. Since they are free from errors due to algebraic manipulations, the convergence is somewhat better for short translations than with the previous methods. This will be shown in Section 6.6.

6.2 Subspace methods

As mentioned in Section 6.1.2 the works of Heeger & Jepson (1992) has led to a number of similar methods being presented by others. A deeper analysis of their approach may thus serve as an introduction to the whole family of such methods. Through algebraic manipulation, the original nonlinear equation (see Equation 6.4) can be split into three sets of equations, where the first set is only related to the translation.

For a given translation $\mathbf{t}$, the translational term of a feature point $\mathbf{x}_i$ may be expressed as $\mathbf{u_t}_i = d_i A_i(\mathbf{t})$, where

$$A_i(\mathbf{t}) = \begin{pmatrix} t_x - x_i t_z \\ t_y - y_i t_z \end{pmatrix} \tag{6.9}$$

and $d_i = 1/Z_i$ is the corresponding inverted depth. The rotational component is given by $\mathbf{u_r}_i = \mathbf{B}(\mathbf{x}_i)\omega$.

With N being the number of vectors, let the optical flow vectors $\mathbf{u}_i$ be stacked on top of each other into a single $2N$-vector $\mathbf{u} = [u_1, v_1, u_2, v_2, .., u_N, v_N]^\top$ and similarly stack the inverted depths together with the rotation parameters, forming $\mathbf{q} = [\, d_1, d_2, .., d_N, \omega_x, \omega_y, \omega_z \,]^\top$. Organize the $A_i(\mathbf{t})$ vectors in a $2N \times N$ bi-diagonal matrix $\mathbf{A}(\mathbf{t})$ and the $\mathbf{B}(\mathbf{x}_i)$ matrices in a single $2N \times 3$ matrix $\mathbf{B}$, that is

$$\mathbf{A}(\mathbf{t}) = \begin{pmatrix} A_1(\mathbf{t}) & 0 & \cdots & 0 \\ 0 & A_2(\mathbf{t}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_N(\mathbf{t}) \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} \mathbf{B}(\mathbf{x}_1) \\ \mathbf{B}(\mathbf{x}_2) \\ \vdots \\ \mathbf{B}(\mathbf{x}_N) \end{pmatrix}. \tag{6.10}$$

Then the optical flow may be expressed as $\mathbf{u} = \mathbf{C}(\mathbf{t})\,\mathbf{q}$, where $\mathbf{C}(\mathbf{t}) = (\mathbf{A}(\mathbf{t})|\mathbf{B})$. A least squares estimate of inverted depths and rotation may thus be found as

$$\hat{\mathbf{q}}(\mathbf{t}) = \mathbf{P}(\mathbf{t})^{-1}\mathbf{C}(\mathbf{t})^{\top}\mathbf{u}, \quad \text{where} \tag{6.11}$$

$$\mathbf{P}(\mathbf{t}) = \begin{pmatrix} \mathbf{A}(\mathbf{t})^{\top}\mathbf{A}(\mathbf{t}) & \mathbf{A}(\mathbf{t})^{\top}\mathbf{B} \\ \mathbf{B}^{\top}\mathbf{A}(\mathbf{t}) & \mathbf{B}^{\top}\mathbf{B} \end{pmatrix}. \tag{6.12}$$

The error of this estimate is given by $e(\mathbf{t}) = |\,\mathbf{u} - \mathbf{C}(\mathbf{t})\,\hat{\mathbf{q}}(\mathbf{t})\,|$, which may be determined without first calculating $\hat{\mathbf{q}}(\mathbf{t})$. Since the error function only depends on the translation, it may be used as an optimization criterion to find the translational estimate $\hat{\mathbf{t}}$. The corresponding inverted depths and rotation will then be readily given by Equation 6.11. Two examples of error functions $e(\mathbf{t})$ can be seen in Figure 6.2 below, with light areas illustrating smaller errors. In the right image double minima can be observed, with a clear extremum representing the "rubber motion" explained in Section 6.1.1.

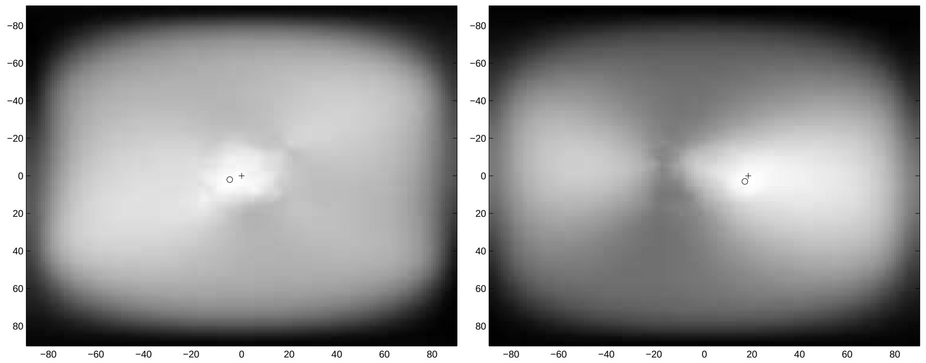

Figure 6.2. The error function $e(\mathbf{t})$ in the cases of two different true translations, $\mathbf{t} = [0, 0, 3]^{\top}$ (left) and $\mathbf{t} = [1, 0, 3]^{\top}$ (right). A cloud of 200 points were generated in a truncated pyramid between 100 and 500 units in front of the cameras, projected onto 320 × 240 pixels images with 0.5 pixels noise.

The elegance of this method is the fact that the search space has been reduced from five dimensions to two, that is those of the translation. However, it relies on an exhaustive search through the space of translations, which might be too costly for real-time use. Calculating $\mathbf{P}(\mathbf{t})^{-1}$ is not as difficult as one might assume, since $\mathbf{A}(\mathbf{t})$ is bi-diagonal and the inversion can be done in O(N) operations. One possibility is using gradient descent from a initial point given by e.g. the 8-point method, that was discussed in Chapter 4. On the other hand, because of its complexity $e(\mathbf{t})$ may contain a whole series of local minima and the 8-point method often yields poor results, especially in the case of confusions between the rotational and translational flows.

However, if the given flow field is dense and the focus of expansion is within the field of view, Srinivasan (2000) has shown that the speed can be greatly improved. Assuming that there are N flow vectors and the focus of expansion is searched for among $M = kN$ locations evenly spread across the image plane, he showed that the computational cost can be reduced from $O(N^2)$ to $O(N \log N)$ using a series of Fast Fourier transforms.

6.2.1 Annihilated rotations

Another method of Jepson & Heeger (1992) tries to first annihilate the rotational component and then solve for the translational direction directly. Thus an exhaustive search can be avoided, hopefully leading to a quicker and more reliable solution. The method can be understood as follows. First a matrix

$$Q(\mathbf{x}_i) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ y_i & -x_i \end{pmatrix} \tag{6.13}$$

is created for each feature point $\mathbf{x}_i$ and multiplied with the corresponding flow vector, that is $\mathbf{q}_i = Q(\mathbf{x}_i)\mathbf{u}_i$. If the image position $\mathbf{x}_i$ is expressed in homogeneous coordinates, it can easily be seen that

$$\mathbf{q}_i = d_i(\mathbf{x}_i \times \mathbf{t}) + \mathbf{x}_i \times (\mathbf{x}_i \times \omega), \tag{6.14}$$

which explains the choice of the matrices $Q(\mathbf{x}_i)$. This means that the rotational term only consists of elements quadratic in feature coordinates, while the translational term is orthogonal to $\mathbf{t}$.

The main idea behind the method is to find a large matrix that eliminates all possible rotational terms that may be defined by $\mathbf{x}_i$. This is possible by creating N-6 N-vectors $\mathbf{c}_k = [\, c_k^1, c_k^2, .., c_k^N]^\top$ that are perpendicular to the 6 N-vectors

$$\{1\},\ \{x_i\},\ \{y_i\},\ \{x_i^2\},\ \{x_i y_i\},\ \{y_i^2\}. \tag{6.15}$$

All elements of $\mathbf{x}_i \times (\mathbf{x}_i \times \omega)$ stacked into N-vectors are sums of such vectors and will also be perpendicular to $\mathbf{c}_k$. Hence all vectors defined by

$$\tau_k = \sum_{i=1}^{N} c_k^i \mathbf{q}_i \tag{6.16}$$

will be orthogonal to $\mathbf{t}$, since the first term of $\mathbf{q}_i$ is already orthogonal to $\mathbf{t}$ and $\mathbf{c}_k$ annihilates all rotational components.

As a consequence $\tau_k^\top \mathbf{t} = 0$ can be used as constraints when searching for a translation, without the rotation being known in advance. The only concern is the fact that the number of constraints have been reduced from N bilinear constraints to N-6 linear ones. An estimate $\hat{\mathbf{t}}$ of the translational direction can thus be found, calculating the least eigenvector of the matrix

$$\mathbf{A} = \sum_{k=1}^{N-6} \tau_k \tau_k^\top . \tag{6.17}$$

Once the translation is known, the rotation and depths can be found as described in the previous section. The major benefit of this method is that it is linear and can be used to initialize a more accurate iterative process. A problem however is that it is prone to serious bias in the translation estimates. This is primarily due to the non-linearity in the projection of image points and the fact that image points are not treated homogeneously. More on the performance of these methods will be presented later on.

6.3 The differential epipolar constraint

As mentioned earlier, solutions based on the epipolar geometry, that was treated in Chapter 4, have serious difficulties in cases of insufficient translation. There is, however, a differential counterpart to the ordinary epipolar constraint and this works on the optical flow rather than on discrete image positions. The constraint was introduced by Zhuang et al. (1988) and later reformulated by Kanatani (1993*a*) in terms of essential parameters and twisted optical flow, that is the flow orthogonal to the projection rays and optical flow.

From Equation 6.2 we already know that the motion of a rigid point can be described as $\dot{\mathbf{X}} = \mathbf{T} + \omega \times \mathbf{X} = \mathbf{T} + \hat{\omega}\mathbf{X}$. Let $\mathbf{x}$ be the projection of $\mathbf{X}$, such that $\mathbf{X} = \lambda\mathbf{x}$. An inner product of the motion equation with $\hat{\mathbf{t}}\mathbf{x}$, where $\mathbf{t} = \mathbf{T}/|\mathbf{T}|$, yields

$$\dot{\mathbf{X}}^\top \hat{\mathbf{t}}\mathbf{x} = (\mathbf{T} + \hat{\omega}\mathbf{X})^\top \hat{\mathbf{t}}\mathbf{x} = (\hat{\omega}\mathbf{X})^\top \hat{\mathbf{t}}\mathbf{x} = \mathbf{X}^\top \hat{\omega}^\top \hat{\mathbf{t}}\mathbf{x} = \lambda \mathbf{x}^\top \hat{\omega}^\top \hat{\mathbf{t}}\mathbf{x}. \tag{6.18}$$

In terms of the projection the motion is $\dot{\mathbf{X}} = \dot{\lambda}\mathbf{x} + \lambda\dot{\mathbf{x}}$. Observing that $\mathbf{x}^\top \hat{\mathbf{t}}\mathbf{x} = 0$, the equation can be rewritten as

$$\dot{\mathbf{X}}^\top \hat{\mathbf{t}}\mathbf{x} = (\dot{\lambda}\mathbf{x} + \lambda\dot{\mathbf{x}})^\top \hat{\mathbf{t}}\mathbf{x} = \lambda\dot{\mathbf{x}}^\top \hat{\mathbf{t}}\mathbf{x}, \tag{6.19}$$

which leads to the Differential Epipolar Constraint

$$\dot{\mathbf{x}}^\top \hat{\mathbf{t}}\mathbf{x} + \mathbf{x}^\top \hat{\omega}\hat{\mathbf{t}}\mathbf{x} = 0. \tag{6.20}$$

Similar to the epipolar constraint, the depths have been eliminated, but it is different in the sense that it works on an image position and its instantaneous optical flow, rather than on image positions from two different images. The matrix $\hat{\omega}\hat{\mathbf{v}}$ is symmetric, which means that the constraint can be reformulated into a symmetric form

$$\dot{\mathbf{x}}^\top \hat{\mathbf{t}}\mathbf{x} + \mathbf{x}^\top \mathbf{A}\mathbf{x} = 0, \text{ where}$$
$$\mathbf{A} = \frac{1}{2}(\hat{\omega}\hat{\mathbf{t}} + \hat{\mathbf{t}}\hat{\omega}), \tag{6.21}$$

which is known as a special symmetric matrix.

The two motion parameters, $\mathbf{A}$ and $\mathbf{t}$, can be found linearly if given at least eight feature points and their corresponding optical flow vectors. However, without any nonlinear optimization there is no guarantee that $\mathbf{A}$ can in fact be decomposed into $\mathbf{t}$ and ω according to Equation 6.21, without imposing any additional constraints. After all, without enforcing decomposability, $\mathbf{A}$ and $\mathbf{t}$ have eight degrees of freedom in total, whereas the constraint only admits five, assuming that $|\mathbf{t}| = 1$. Ma et al. (2000) described a scheme in which $\mathbf{A}$ is instead expressed in terms of ω and a new vector $\mathbf{t}'$. The alternative, $\mathbf{t}$ or $\mathbf{t}'$, that results in the lowest residual is then chosen as the final translational direction. They also presented a four-step algorithm for estimating the motion, similar to the three-step SVD-based method used for essential matrices proposed by Toscani & Faugeras (1986). The method is summarized below.

Algorithm

1. Let $\mathbf{y}_i = (x_i^2, 2x_iy_i, 2x_i, y_i^2, 2y_i, 1, -v_i, u_i, v_ix_i - u_iy_i)^\top$ and the optical flow be defined by $\dot{\mathbf{x}}_i = (u_i, v_i, 0)^\top$. Find the vector $\mathbf{a} = (a_1, a_2, a_3, a_4, a_5, a_6, t_x, t_y, t_z)^\top$ that minimizes $\sum_i (\mathbf{a}^\top \mathbf{y}_i)^2$, where the motion parameters are

$$\mathbf{A}_0 = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_2 & a_4 & a_5 \\ a_4 & a_5 & a_6 \end{pmatrix} \quad \text{and} \quad \mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}. \tag{6.22}$$

2. Perform an eigenvalue decomposition of $\mathbf{A}_0$, $\mathbf{A}_0 = \mathbf{V}_0 \; diag\{\lambda_1, \lambda_2, \lambda_3\}\mathbf{V}_0^\top$, with the eigenvalues ordered as $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Create a new matrix through a projection of $\mathbf{A}_0$ into the space of special symmetric matrices,

$$\mathbf{A} = \mathbf{V}_0 \; diag\{\sigma_1, \sigma_2, \sigma_3\} \; \mathbf{V}_0^\top. \tag{6.23}$$

   The eigenvalues of the new matrix are defined as $\sigma_1 = \frac{1}{3}(2\lambda_1 + \lambda_2 - \lambda_3)$, $\sigma_2 = \frac{1}{3}(\lambda_1 + 2\lambda_2 + \lambda_3)$ and $\sigma_3 = \frac{1}{3}(2\lambda_3 + \lambda_2 - \lambda_1)$.

3. Define the rotational magnitude $\lambda = \sigma_1 - \sigma_3$ and an angle $\theta = \arccos(-\sigma_2/\lambda)$. With $\mathbf{R}_\mathbf{y}(\theta)$ being a rotation around the y-axis, let

$$\mathbf{U} = \mathbf{V}_0\mathbf{R}_\mathbf{y}(\frac{\pi}{2} + \frac{\theta}{2}), \quad \mathbf{V} = \mathbf{V}_0\mathbf{R}_\mathbf{y}(\frac{\pi}{2} - \frac{\theta}{2}) \quad \text{and} \tag{6.24}$$

$$\mathbf{Z} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{6.25}$$

   Four possible rotations and translational directions are then given by

$$\begin{cases} \hat{\omega} = \pm\lambda\mathbf{U}\mathbf{Z}\mathbf{U}^\top, & \hat{\mathbf{t}}' = \pm\mathbf{V}\mathbf{Z}\mathbf{V}^\top \\ \hat{\omega} = \pm\lambda\mathbf{V}\mathbf{Z}\mathbf{V}^\top, & \hat{\mathbf{t}}' = \pm\mathbf{U}\mathbf{Z}\mathbf{U}^\top \end{cases} \tag{6.26}$$

4. Choose the rotation ω that corresponds to the estimate $\mathbf{t}'$ closest to $\mathbf{t}$ given in Step 1 and then either $\mathbf{t}'$ or $\mathbf{t}$ depending on which translation gives rise to the lowest residual error.

Proofs concerning the decomposition of $\mathbf{A}$ can be found in (Ma et al. 2000). As can be seen in Step 3, there are four alternative solutions. However, like with the essential matrix, ambiguous solutions can be thrown away using the positive depth constraint. The main reason for using the differential version of the epipolar constraint is that the discrete variant tends to perform badly for small motions. Since the differential one is based on instantaneous optical flow, it is likely to perform better. In the next section we will see if that is the case.

6.4 Experiments

Even if a higher accuracy in general can be achieved using nonlinear methods, such methods typically require an initial motion estimate close enough to the true solution. In this section we therefore intend to evaluate three methods that could initiate such a process. A series of simulations were performed using randomly generated data. This generation of data, which will be described later on in this section, was done using parameters representing that of typical working conditions for an autonomous platform moving around in an indoor environment. The performance is evaluated in terms of 3D velocities, rather than magnitudes of the optical flow. This means that the conclusions might be slightly different from those of other studies.

The first method to be analyzed is the usual 8-point method, that was discussed in Chapter 4 and tested for stereo. Motion is different because the epipolar points are typically located within the image plane, which only occurs in stereo for very asymmetric configurations. As mentioned earlier, problems may arise due to the so called second eigenmotion, that is the motion due to the second smallest eigenvalue of the linear least squares solution, explained in Section 4.4.1. If the points are not properly spread in depth and field of view and if the noise level is too high, the ordering of eigenvectors may change, resulting in an incorrect local minimum being found.

The reason for this instability of the linear method, is partially that the error function has not been properly normalized. This typically leads to a bias towards translations along the optical axis. In order to cope with this weakness, an additional nonlinear search is performed in the one-dimensional space defined by the essential matrices $\mathbf{E_{9}}_1$ and $\mathbf{E_{9}}_2$, that are associated with the two least eigenvalues. For essential matrices defined by $\mathbf{E}(\alpha) = \cos(\alpha)\, \mathbf{E_{9}}_1 + \sin(\alpha)\, \mathbf{E_{9}}_2$ and $\mathbf{z} = (0,0,1)^\top$, the following normalized error function is minimized,

$$f_d(\alpha) = \sum_{\mathbf{x_l},\mathbf{x_r}} \frac{(\mathbf{x_l}^\top \mathbf{E}(\alpha)\mathbf{x_r})^2}{(\hat{\mathbf{z}}\mathbf{E}(\alpha)\mathbf{x_r})^2 + (\mathbf{x_l}^\top \mathbf{E}(\alpha)\hat{\mathbf{z}})^2}. \tag{6.27}$$

The next method to be studied is the one based on the instantaneous epipolar constraint, that was described in Section 6.3. Unfortunately, the same problems exist in this case and the corresponding error function should also here be normalized. With $\mathbf{t}(\alpha)$ and $\mathbf{A}(\alpha)$ defined like $\mathbf{E}(\alpha)$ given above, the optimal motion is found minimizing

$$f_i(\alpha) = \sum_{\mathbf{x}} \frac{(\dot{\mathbf{x}}^\top \hat{\mathbf{t}}(\alpha)\mathbf{x} + \mathbf{x}^\top \mathbf{A}(\alpha)\mathbf{x})^2}{(\hat{\mathbf{z}}\hat{\mathbf{t}}(\alpha)\mathbf{x})^2}. \tag{6.28}$$

The nonlinear optimizations are performed using Levenberg-Marquardt (Press et al. 1992), starting with the eigenmotion of lowest normalized error. Since the search is performed in one dimension only, a minimum is reached in about 10 iterations. The last method analyzed in this section is the subspace method of Jepson & Heeger (1992) described in Section 6.2.1. The reason for choosing these three methods, is that they all work without any prior information and can thus be used to initialize more accurate iterative methods.

Simulated data Based on what has shown to be typical for the system presented in this thesis, series of 80 feature pairs each were randomly generated in a truncated pyramid 200 to 600 cm away from an imaginary observer and projected onto two 384 × 288 pixel image planes. The focal length was set to 400 pixels, which corresponds to a field of view of about 52°. Noise with a standard deviation of 0.7 pixels was added to each image feature position. However, unlike a real scenario, all features pairs were assumed to be correctly matched, which means that no outliers were included in the data sets. Thus considerable additional work has to be done to guarantee robustness, if the methods are to be used in practice, similar to what is described in Chapter 4 in the case of stereo.

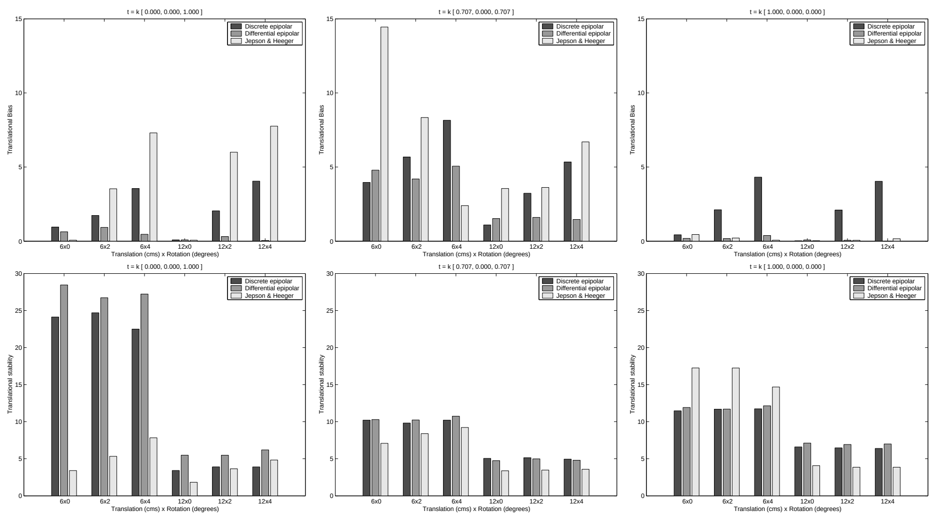

Figure 6.3. Translational bias (upper) and stability (lower), for motions along the optical axis (left), diagonally (middle) and along the x-axis (right). Each group of bars show the results for different combinations of translational and rotational speeds, with rotations around the y-axis.

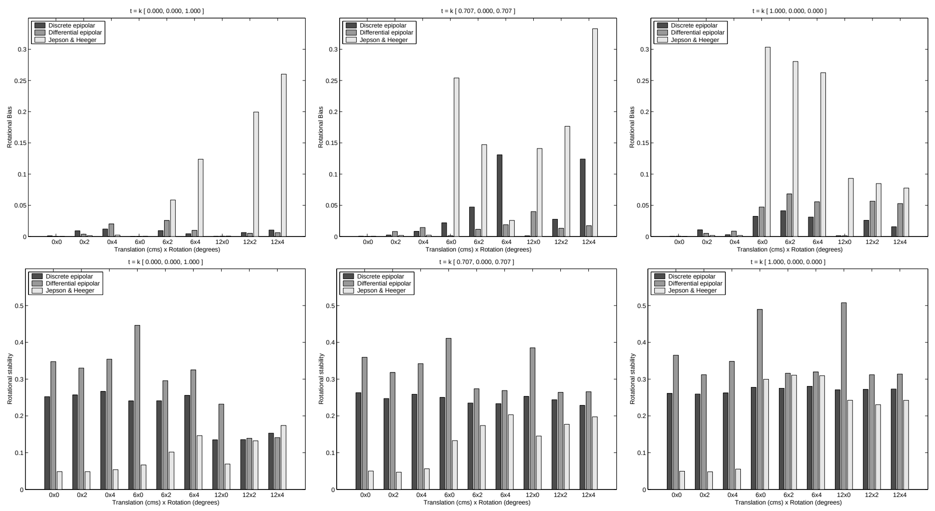

Figure 6.4. Rotational bias (upper) and stability (lower), for motions along the optical axis (left), diagonally (middle) and along the x-axis (right). Each group of bars show the results for different combinations of translational and rotational speeds, with rotations around the y-axis.

Simulations were performed with translations in three different directions, along the optical axis, the x-axis and the diagonal in the plane defined by these two axes. All rotations were around the y-axis, making the rotational flow hard to distinguish from the translational one, which was explained in Section 6.1.1. Rotations around other axes have previously shown to have little effect on the stability of the problem. If an observed object is kept fixated, while the observer is moving about, the cameras are rotated so as to subtract the flow induced by the translation. Hence, these motions are common for autonomous robots.

The performance of the three methods was evaluated in terms of bias and stability, using a metric proposed by Tian et al. (1996). With 1000 different data sets being tested for each combination of true rotation and translation, a mean motion was calculated. The corresponding biases can be seen in the upper rows of Figures 6.3 and 6.4. The lower rows show the stability of the results, which was calculated as the variance using the same metric. Note that longer bars indicate higher variance and thus less stable results. The results of the discrete epipolar constraint, the differential epipolar constraint and the method of Jepson & Heeger are shown in black, grey and white respectively.

Compared to most other studies the translations were relatively small, which greatly complicates the estimation problem. However, a maximum magnitude of 12 cm, that was used here, is equivalent to as much as 3 m/s if the system runs at 25 Hz, or 1 m/s when only every third frame is used. On the other hand, a longer delay between image frames means an increased risk of outliers due to independent motion and a longer latency between events occurring in the scene and the observer being able to react. For example, if ego-motion is used for calculating time-to-contact, this latency is indeed critical.

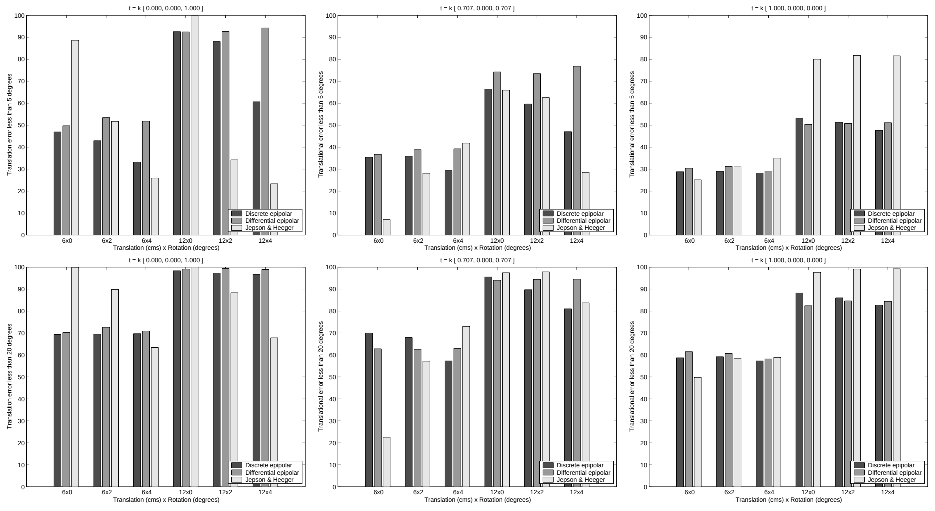

Figure 6.5. Fraction of estimates with errors in translational direction less that 5° (upper row) and 10° (lower). Each group of bars show the results for different translational and rotational speeds, with rotations around the y-axis.

The results show that the difference between the discrete epipolar constraint and the differential one is primarily in the bias and not in stability. It should be pointed out that the additional nonlinear optimizations greatly improved the bias in both cases, but at the cost of a somewhat worsened stability. The subspace method shows a considerably better stability, especially for smaller translations, but the bias can in some cases be very large. The smallest bias was observed using the differential epipolar constraint method. It has been shown (Daniilidis & Nagel 1990), that motions along the optical axis yield more stable results than those parallel to the image plane. This can be seen if the translational magnitudes are increased. However, from the results in Figure 6.3 this is not at all obvious. For small forward translations the induced flow is so small, that it is hard to discriminate it from ordinary image noise. This leads to very large translational variances, which is especially evident in the epipolar constraint based methods.

In conclusion, there is no clear winner among the three algorithms. If the choice is between the two epipolar constraint methods, the differential one seems to be the preferred choice. For some applications the type of motion might be more important than the exact measures. A complete solution might in fact consist of a number of methods, one method dedicated for each motion type. An approach to determine which method to use would then be required. This approach would most likely be successful, even if the bias is significant. In such a case, the subspace method might be more favourable.

If any of the evaluated methods is to be used to initialize a more accurate iterative one, it is important to know if the initial motion estimates are close enough to the true solution. In Figure 6.5 the fraction of estimates with translational errors less that 5° and 10° are shown in the upper and lower row respectively. As long as the iterative method is able to converge, the initial estimates need only to be good enough. Rather disappointingly the best method, which seems to be the one based on the differential epipolar constraint, is only able to guarantee 30% of estimates being closer than 5° to the true translational direction. If an error of 10° is acceptable, the fraction increases to about 60%. However, this fraction decreases rapidly if the translational magnitude is less than 6 cm. The question is whether the results are good enough for an iterative method to be initialized. The success depends on the ability of such a method to converge, which will be analyzed in the following section.

6.5 Iterative methods

The original optimization problem using the bilinear constraint is nonlinear, as stated in Section 6.1. In order to solve the problem most methods try to eliminate some parameters and then determine the remaining ones linearly. Unfortunately, this process typically leads to bias, when the error functions are manipulated algebraically. Worse yet, due to the interdependence between parameters, a bias in one parameter will affect the others. Thus the nonlinear problem ought to be solved iteratively in order to reach a result without such a bias. In this section three different nonlinear methods will be described. Each method relies on an initial motion estimate being close enough to the true motion, but they differ in the way bias is eliminated.

6.5.1 Annihilated rotations

Oliensis & Genc (1999) proposed an algorithm that like the method of Jepson & Heeger (1992) tries to annihilate the rotational component and then optimize for translation. However, the annihilation is not performed on the optical flow directly, but on the flow component perpendicular to the translational one. Thus given a correct translational direction, the results after annihilation should be just noise. The benefit of this approach is that annihilation is performed using three N-vectors only, instead of six, and as a consequence of this it is able to better handle planar scenes. The reason why Jepson & Heeger (1992) have problems with planar scenes is that the motion due to a plane can be expressed in terms of the annihilated vectors of Equation 6.15.

The three annihilation vectors correspond to the rotational components projected such that they are perpendicular to the current translational flow. The vectors are collected in a $N \times 3$ matrix of the form

$$\mathbf{V}_k = [\,\{\tau_k(\mathbf{x}_i) \times \mathbf{r}_i^x\}, \{\tau_k(\mathbf{x}_i) \times \mathbf{r}_i^y\}, \{\tau_k(\mathbf{x}_i) \times \mathbf{r}_i^z\}\,]\,, \text{ where} \tag{6.29}$$

$$\tau_k(\mathbf{x}_i) = \frac{A(\mathbf{x}_i)\mathbf{t}_k}{|A(\mathbf{x_i})\mathbf{t}_k|} \tag{6.30}$$

and $\mathbf{r}_i^x$, $\mathbf{r}_i^y$ and $\mathbf{r}_i^z$ are the three columns of the rotational flow basis $B(\mathbf{x}_i)$ in Equation 6.6. The algorithm of Oliensis & Genc (1999) is given below.

Algorithm

1. Let $k = 0$ and find an initial estimate of the translation $\mathbf{t}_0$ and rotation ω_0 using e.g. the 8-point method.
2. Compensate for the currently known rotational flow and update the translation $\mathbf{t}_{k+1} = \arg\min_{\mathbf{t}} |\mathbf{V}_k^{\perp}\boldsymbol{\Upsilon}_k|^2$, with
$$\boldsymbol{\Upsilon}_k = \{\, \tau_k(\mathbf{x}_i) \times (\mathbf{u}_i - B(\mathbf{x}_i)\,\omega_k)\}.$$
3. Update the annihilation matrix $\mathbf{V}_{k+1}$.
4. Calculate a new rotational estimate of $\omega_{k+1} = \omega_k + \Delta\omega_k$, where the update is determined by $\Delta\omega_k = \arg\min_{\Delta\omega} |\boldsymbol{\Upsilon}_k - \mathbf{V}_{k+1}\Delta\omega|^2 = \mathbf{V}_{k+1}^{\dagger}\boldsymbol{\Upsilon}_k$.
5. Return to Step 2 until convergence.

The minimization of Step 2 can be performed through steepest descents, using for example the method of Levenberg-Marquardt. To avoid getting stuck in a local minimum, such as one corresponding to the "rubber motion", a search can initially be performed among directions located on the plane of the current translation $\mathbf{t}_k$ and the optical axis.

6.5.2 Simultaneous structure and motion

The main reason for the bias in the previous methods is the algebraic manipulations done in order to divide the problem into different components that are estimated separately. In order to avoid computing depths these methods based their residuals on the component orthogonal to the translational flow, completely ignoring depth. In the method of Zhang & Tomasi (1999) the depths are instead estimated simultaneously and the minimization is performed on the original error function, that is their method tries to minimize the residual

$$\mathbf{r}_i(d_i, \mathbf{t}, \omega) = \mathbf{u}_i - d_i A(\mathbf{x}_i)\,\mathbf{t} - B(\mathbf{x}_i)\,\omega. \tag{6.31}$$

The translational direction, rotation and depths are estimated in sequence, using the same error function, with total least squares being used in each step, except for the step updating the rotation. Instead rotational estimates are based on the flow orthogonal to the current translational one. However, for translations in the forward direction, rotations based on errors in both dimensions seem to result in a significantly better convergence rate. Despite this the original algorithm, which is given below, was used for the simulations presented in Section 6.6.

Algorithm

1. Let $k = 0$, the inverted depths $d_{i,0} = 1$ and find an initial estimate of the translation $\mathbf{t}_0$ and rotation ω_0 using e.g. the 8-point method.
2. Calculate an updated translational direction based on the current residual,

$$\Delta\mathbf{t}_k = \arg\min_{\Delta\mathbf{t}} \sum_{i=1}^{N} |\mathbf{r}_i(d_{i,k-1}, \mathbf{t}_{k-1}, \omega_{k-1}) - d_i A(\mathbf{x}_i)\Delta\mathbf{t}|^2 \tag{6.32}$$

   with the condition $\mathbf{t}_{k-1}^\top \Delta\mathbf{t}_k = 0$ and update as $\mathbf{t}_k = \mathbf{t}_{k-1} + \Delta\mathbf{t}_k$.
3. Like the method of Oliensis & Genc (1999) and with definitions in Section 6.5.1, the rotation is updated as $\omega_k = \arg\min_\omega |\boldsymbol{\Upsilon}_k - \mathbf{V}_k\omega|^2 = \mathbf{V}_k^\dagger \boldsymbol{\Upsilon}_k$.
4. Individually for each feature point, recompute the depths,

$$d_{i,k} = \arg\min_{d_i} |\mathbf{r}_i(d_i, \mathbf{t}_k, \omega_k)|^2 = \frac{\tau_k^\top(\mathbf{x}_i)(\mathbf{u}_i - B(\mathbf{x}_i)\,\omega_k)}{\tau_k^\top(\mathbf{x}_i)A(\mathbf{x}_i)\mathbf{t}_k}. \tag{6.33}$$

5. Return to Step 2 until convergence.

6.5.3 Spherical projections

The methods studied previously in this thesis have been based on perspective projections. However, optical flow expressed in spherical projections leads equations, which can sometimes be easier to understand and analyze (Thomas et al. 1994, Soatto & Perona 1997). Using spherical coordinates, that is $\mathbf{x}_i \in \mathbf{S}^2$, the optical flow can be written as

$$\mathbf{u}_i = d_i(\mathbf{t} \times \mathbf{x}_i) \times \mathbf{x}_i + \omega \times \mathbf{x}_i = d_i\hat{\mathbf{x}}_i^2\mathbf{t} - \hat{\mathbf{x}}_i\omega. \tag{6.34}$$

Thus the optical flow may be considered as the result of a rotation around an axis $\omega_i = \omega - d_i\hat{\mathbf{x}}_i\mathbf{t}$, which is unique for every flow vector. The flow can further be measured against the image point vectors, which yields the angular flow

$$\mathbf{y}_i = \mathbf{u}_i \times \mathbf{x}_i = -d_i\hat{\mathbf{x}}_i\mathbf{t} + \hat{\mathbf{x}}_i^2\omega. \tag{6.35}$$

It should be emphasized that if the image noise is isotropic in the spherical projection, it will be so also after this manipulation. In an algorithm presented by Chiuso et al. (2000), the residual

$$r(d_i, \omega, \mathbf{t}) = \sum_{i=0}^{N} |\mathbf{y}_i + d_i\hat{\mathbf{x}}_i\mathbf{t} - \hat{\mathbf{x}}_i^2\omega|^2 \tag{6.36}$$

is minimized, which is done linearly for each component. Instead of enforcing the translational magnitude to be $|\mathbf{t}| = 1$, normalization is done after optimization. Based on experimental results there does not seem to be much difference between the two alternative procedures.

Using an initial zero rotation and all points set to some arbitrary positive depth, the parameters are searched iteratively, first estimating the translation, then the rotation and finally the depths. The procedure is performed twice, but in the second run the initial values of Step 1 are changed to that of the corresponding "rubber motion", that is $\omega_0 = -\omega$ and $d_{i,0} = 2\bar{d} - d_i$. An additional check is performed on the inverted depths making sure that they remain positive. Negative depths are changed to random values and the procedure is continued. After convergence the result with the lowest total residual is used as the final estimate of motion and structure. The complete algorithm is as follows.

Algorithm

1. Let $k = 0$, $\omega_0 = 0$ and $d_{i,0} = 1$.
2. Compute $\mathbf{t'}_k = \arg\min_{\mathbf{t'}} r(d_{i,k-1}, \omega_{k-1}, \mathbf{t'})$ as
$$\mathbf{t'}_k = (\sum_{i=0}^{N} d_{i,k-1}^2 \hat{\mathbf{x}}_i^2)^{-1} \sum_{i=0}^{N} d_{i,k-1} \hat{\mathbf{x}}_i (\omega_{k-1} - \mathbf{y}_i).$$
3. Normalize the translation $\mathbf{t}_k = \mathbf{t'}_k / |\mathbf{t'}_k|$.
4. Update the rotation,
$$\omega_k = \arg\min_{\omega} \sum_{i=0}^{N} |\mathbf{x}_i \mathbf{t}_k^\top (\mathbf{y}_i - \hat{\mathbf{x}}_i^2 \omega)|^2 = (\sum_{i=0}^{N} \hat{\mathbf{x}}_i^2 \mathbf{t}_k \mathbf{t}_k^\top \hat{\mathbf{x}}_i^2)^{-1} \sum_{i=0}^{N} \hat{\mathbf{x}}_i^2 \mathbf{t}_k \mathbf{t}_k^\top \hat{\mathbf{y}}_i.$$
5. Individually recompute the inverted depths through
$$d_{i,k} = \arg\min_{d_i} r(d_i, \omega_k, \mathbf{t}_k) = (\hat{\mathbf{x}}_i^2 \omega_k - \mathbf{y}_i)^\top \frac{\hat{\mathbf{x}}_i \mathbf{t}_k}{\mathbf{t}_k^\top \hat{\mathbf{x}}_i^2 \mathbf{t}_k} \qquad (6.37)$$
6. Return to Step 2 until convergence.

6.6 Experiments

The three iterative algorithms in Section 6.5 were tested on simulated data generated as in Section 6.4. However, even if the previously presented linear methods were intended for initialization, the iterative procedures were initialized using a cloud of random initial translations and rotations. Thus Step 1 of the given algorithms was changed so as to better analyze the convergence of the three methods. The initial motion estimates were generated such that translational directions were evenly spread within a quadratic angular area with sides of length 20° in altitude as well as in azimuth.

In order to facilitate convergence as many as 32 iterations were used for each algorithm, which is more than what some of the algorithm designers themselves recommended. The number of resulting samples with a translational direction within 5° and 10° of the true one, can be seen in Figure 6.6. Before drawing any

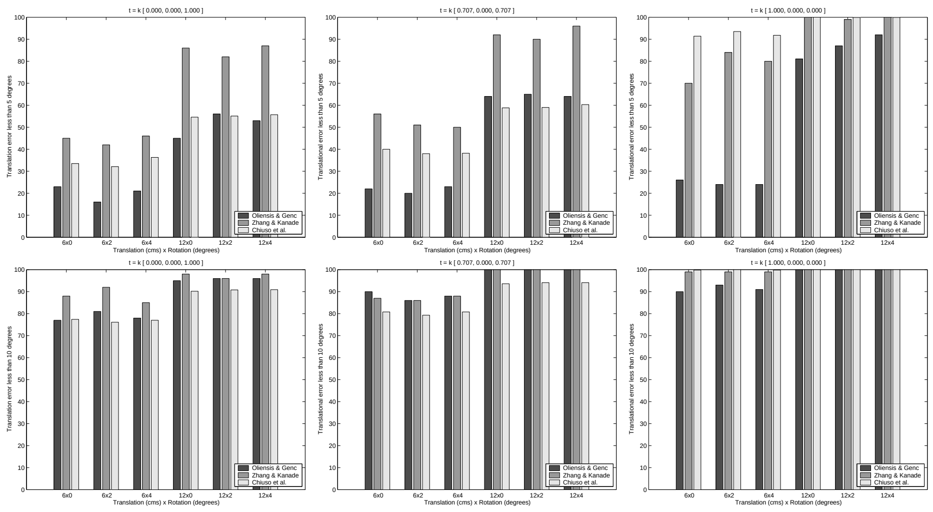

Figure 6.6. Fraction of estimates with errors in translational direction less that 5° (upper row) and 10° (lower). Each group of bars show the results for different translational and rotational speeds, with rotations around the y-axis.

conclusions one should be reminded that out of the initial randomized motion estimates, about 19.6% have a translational error of less than 5° and 78.5% are within 10°. Thus the results are not as good as one might initially assume.

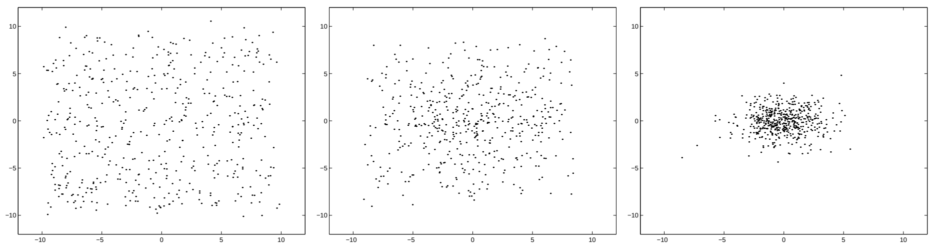

Figure 6.7. Distribution of translational results for magnitudes of 6 cm (left), 12 cm (centre) and 18 cm (right), using the method of Oliensis & Genc.

In the simulations, data were generated such that depths and image positions were uncorrelated. This makes the expected standard deviation of displacements due to translations relatively easy to calculate. Using the simulated data given in Section 6.4, it is possible to show that the approximate displacement in the forwards direction is about $\text{std}(\mathbf{u_t}) = 0.19\, t_z$ pixels, with t_z measured in centimetres. This means that 6 cm in forward translation is equivalent to a noise level of about 44.5%.

Such a short translation is obviously not enough to reach a proper convergence. One exception is in the lateral direction, where the two methods that include depth estimates are able to converge. The situation is quite different if the translations are increased to 12 cm, where all methods converge more or less, depending on the direction. The most successful approach is that of Zhang & Kanade, which converges quite well even in the forward direction. Rotations do not seem to affect convergence at all. Another way of analyzing the convergence is by studying the images of Figure 6.7. For the method of Oliensis & Genc, it is not until the magnitude is as large as 18 cm that convergence is clearly visible.

It should be pointed out that the reported results depend on the number of iterations used. The method of Oliensis & Genc shows significant improvements if the number of iterations is increased. However, even with as many as 100 iterations the results do not meet those of the other methods. An effect of the annihilation seems to be that the problem becomes more sensitive to noise. Annihilations are performed to isolate residuals due to certain sets of motion parameters. On the other hand, if the annihilation is not perfect the residual contains not just image noise, but also noise due to the erroneous annihilation. Furthermore, even if convergence is possible, the high number of local extrema in the forward direction further complicates the process (Oliensis 2000). These exist due to the denominator of the error function mentioned is Section 6.1.1, which is similar for all the reviewed methods.

Since the convergence is so slow for all the three tested iterative methods, at least for our application, we have chosen not to use such an approach in the system presented in Chapter 8. We will instead rely on data from binocular stereo. It should be emphasized that even if the convergence were better, one still has to include a mechanism for outlier detection. From the discussions in Chapter 4 one may conclude that outliers would further complicate the problem.

6.7 Conclusions

In this chapter a number monocular structure-from-motion methods have been analyzed in the context of an autonomous system moving around in an indoor scene. The intention was to determine whether such a method may be used for ego-motion estimation. Three linear and three iterative methods were evaluated. The linear methods are important, since they may be used to initialize a more accurate iterative one. All methods show similar weaknesses for translations shorter than about 6 cm. Since the convergence of the iterative methods was so weak, in relation to the results of the linear ones, we judged them not feasible for our particular application.

In conclusion, since the translations in an indoor scene are typically short in relation to what the evaluated structure-from-motion methods require, we decided not to rely on such a method. The situation might, however, be much different in an application such as reconstruction. In the next chapter we will instead investigate the possibility of using stereo data for ego-motion estimation.

Chapter 7

Stereo and motion

The results of the previous chapter were not as encouraging as one would have hoped, even if monocular structure-from-motion is feasible as long as the translational speed is high enough. On the other hand, if the autonomous system is equipped with a binocular stereo head, reconstructed three-dimensional feature points might be available from stereo. We claim that the process of determining ego-motion can be simplified, when such points exist. In this chapter the use of stereo for ego-motion estimation will be explored, beginning with an analysis of the errors involved in the reconstruction of 3D features. A number of alternative methods for estimating ego-motion will be given. In the end of the chapter, ego-motion will be combined with disparities, calculated as described in Chapter 5, resulting in a system in which independently moving objects can be found quickly and automatically.

7.1 Triangulation

Three dimensional feature points may be created though a process called triangulation, given that image features have been properly matched and the epipolar geometry is known. Assume that we have a binocular stereo system and use the coordinate system given in Section 4.3. The origin is defined by the left camera centre and both optical axes lie on the same plane, which also contains the baseline $\mathbf{b} = (b, 0, 0)^\top$. Let a 3D point $\mathbf{p}$ be projected onto the left and right image planes. The projection can be represented by two normalized rays, $\mathbf{x_l} = (x_l, y_l, z_l)^\top$ and $\mathbf{x_r} = (x_r, y_r, z_r)^\top$. With s_l and s_r denoting the distances from each camera centre to $\mathbf{p}$, the position of $\mathbf{p}$ can be given by either $\mathbf{p_l}(s_l) = s_l\mathbf{x_l}$ or $\mathbf{p_r}(s_r) = s_r\mathbf{x_r} + \mathbf{b}$. Equating these two vectors, $s_l\mathbf{x_l} = s_r\mathbf{x_r} + \mathbf{b}$, yields the equation

$$\left(\mathbf{x_l}, -\mathbf{x_r} \right) \begin{pmatrix} s_l \\ s_r \end{pmatrix} = \mathbf{b}. \tag{7.1}$$

The distances can be found using least squares, which results in

$$\begin{pmatrix} |\mathbf{x_l}|^2 & -\mathbf{x_l}^\top \mathbf{x_r} \\ -\mathbf{x_r}^\top \mathbf{x_l} & |\mathbf{x_r}|^2 \end{pmatrix} \begin{pmatrix} s_l \\ s_r \end{pmatrix} = b \begin{pmatrix} x_l \\ -x_r \end{pmatrix}. \tag{7.2}$$

Observing that $|\mathbf{x_l}| = |\mathbf{x_r}| = 1$ and $\mathbf{x_l}^\top \mathbf{x_r} = \cos(\beta)$, where β is the angle between the projection rays $\mathbf{x_l}$ and $\mathbf{x_r}$, the distances are given by

$$\begin{pmatrix} s_l \\ s_r \end{pmatrix} = \frac{b}{\sin^2(\beta)} \begin{pmatrix} 1 & \cos(\beta) \\ \cos(\beta) & 1 \end{pmatrix} \begin{pmatrix} x_l \\ -x_r \end{pmatrix}. \tag{7.3}$$

If $\mathbf{p}$ is located relatively far away from the stereo head compared to the length of the baseline and if the camera configuration is close to symmetric, a number of approximations can be introduced. Then $\sin(\beta) \approx \beta \approx x_l - x_r$ and Equation 7.3 can be simplified as

$$\begin{pmatrix} s_l \\ s_r \end{pmatrix} \approx \frac{b}{\beta^2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_l \\ -x_r \end{pmatrix} \approx \frac{b}{\beta} \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{7.4}$$

A symmetric configuration is not just motivated from a computational point of view, such as indicated by the results in Chapter 5. The cameras can never rotate more than what is determined by the mechanics. In order for the observer to execute a saccade towards any position within the field of view, the cameras should typically be directed forwards, in relation to the neck on which the cameras are mounted. During and after a saccade, eye movements could be compensated by a synchronized rotation of the neck (Pahlavan et al. 1993). Thus the stereo system will stay close to symmetric between two saccades.

The distances can be regarded as depths viewed from respective camera. As mentioned in Chapter 6, these depths constitute the structure in the feature based structure-from-motion problem. Thus knowing the dependency between depths and image noise is essential in order to determine the expected quality of the extracted structure. Due to image noise the angle β will be erroneous. Since the depths are given by $Z = b/\beta$, the reconstruction errors depend on the derivative

$$\frac{\delta Z}{\delta \beta} = -\frac{b}{\beta^2} = -\frac{Z^2}{b} \tag{7.5}$$

or if expressed using inverted depths $Z^{-1} = \beta/b$,

$$\frac{\delta Z^{-1}}{\delta \beta} = \frac{1}{b}. \tag{7.6}$$

Thus there is a quadratic dependency between depths and the errors of the reconstruction, while the inverted depths only depend on the baseline. This quadratic dependency can be illustrated by the image in Figure 7.1. The shaded area represents a region of uncertainty. As the depth is increased, the height H of the uncertainty region grows quadratically.

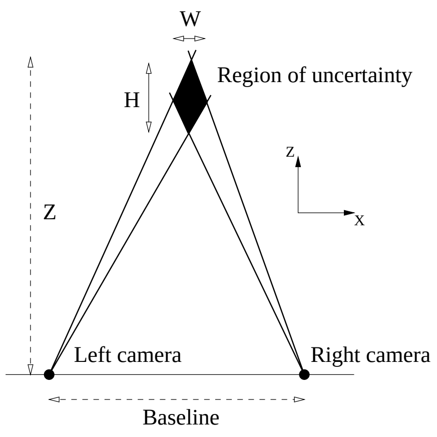

Figure 7.1. Region of uncertainty depending on image point errors.

How much do these errors affect the performance of a practical application, such as for example manipulation? If the working distance is about one metre and the image errors are in the neighbourhood of one pixel, with a focal length equivalent to 400 pixels and a baseline of 20 cm, it is easily seen that the errors in reconstructed depths are about 1.25 cm. For indoor navigation, where the distances are typically longer, the errors can be of the order of decimetres and metres. Unfortunately, image point errors are not the only errors involved. Additional errors originate from difficulties in determining the camera calibration. However, even if the absolute errors in depth are large, the relative ones may still be manageable. Thus these errors may well be overcome, if the manipulator arm is visible and tracked.

7.2 Ego-motion using depths

If our observing agent is equipped with a binocular stereo head, one might wonder if a more accurate estimate of ego-motion can be achieved using two cameras, instead of just one. After all, the errors in depth depend on the length of the baseline and typically the baseline is longer than the translational magnitude. The fact that two images from the left and right cameras originate from the same instance in time implies that the number of outliers, due to non-rigid motion, can be expected to be much fewer. The structure-from-motion problem is in itself very error sensitive and every single inlier is valuable. However, an efficient exclusion of outliers often results in some correct matches being falsely disregarded. Thus the problem would benefit not just from the longer baseline, but also from a fewer number of outliers being present.

Assume that a number of features have been matched between the left and right cameras, and that the corresponding depths have been found using triangulation. One possible way of estimating the ego-motion could then be using these depths in a structure-from-motion algorithm, e.g. one of those described in Section 6.5. A good candidate might be the method of Zhang & Tomasi (1999) reviewed in Section 6.5.2, since in their approach the depths are not eliminated, unlike methods based on the essential matrix. In this section of the thesis we instead use an error function, where an a priori depth estimate has been incorporated. The error associated with a feature at image position $\mathbf{x}_i$ is given by

$$e_i(\omega, \mathbf{t}, d_i) = |\mathbf{u}_i - d_i A(\mathbf{x}_i)\mathbf{t} - B(\mathbf{x}_i)\omega|^2 + K|d_i - d_i^s|^2, \tag{7.7}$$

where d_i and d_i^s denote the estimated and a priori given inverted depths and K determines the relative importance of d_i^s. The remaining variables are given in Section 6.1. A differentiation of $e_i(\omega, \mathbf{t}, d_i)$ with respect to depth yields

$$\frac{\delta}{\delta d_i} e_i(\omega, \mathbf{t}, d_i) = 2(A(\mathbf{x}_i)\mathbf{t})^\top(\mathbf{u}_i - d_i A(\mathbf{x}_i)\mathbf{t} - B(\mathbf{x}_i)\omega) + 2K(d_i - d_i^s), \tag{7.8}$$

which leads to an estimated depth given by

$$\hat{d}_i = \frac{K d_i^s + (A(\mathbf{x}_i)\mathbf{t})^\top(\mathbf{u}_i - B(\mathbf{x}_i)\omega)}{K + (A(\mathbf{x}_i)\mathbf{t})^\top A(\mathbf{x}_i)\mathbf{t}}. \tag{7.9}$$

The ego-motion can be found using Equation 7.9 in Step 4 of the iterative algorithm described in Section 6.5.2. Due to the ambiguity between depths and translational magnitude, the translation $\mathbf{t}$ was normalized in Step 2 of that algorithm. Since depths are known a priori this normalization is no longer necessary and the condition that $\Delta\mathbf{t}_{k-1}^\top \mathbf{t}_k = 0$ can be ignored.

7.2.1 Initial motion estimates

Unfortunately, most iterative methods rely on good initial estimates in order to be successful and the experiments in Section 6.4 indicated that such estimates are very difficult to find, especially if the translations are small. The speed of an autonomous platform in an indoor environment is typically too low, compared to what is required. If the depths are already available from stereo, the situation might be slightly different. Then it is possible to directly solve the rotation and translation linearly. If the optical flow due to ego-motion is expressed as

$$\mathbf{u}_i = \left(d_i A(\mathbf{x}_i) \mid B(\mathbf{x}_i) \right) \begin{pmatrix} \mathbf{t} \\ \omega \end{pmatrix}, \tag{7.10}$$

a motion estimate can be found using least squares, that is

$$\begin{pmatrix} \hat{\mathbf{t}} \\ \hat{\omega} \end{pmatrix} = \mathbf{S}^{-1} \begin{pmatrix} \sum_i d_i A(\mathbf{x}_i)^\top \mathbf{u}_i \\ \sum_i B(\mathbf{x}_i)^\top \mathbf{u}_i \end{pmatrix}, \text{ where} \tag{7.11}$$

$$\mathbf{S} = \begin{pmatrix} \sum_i d_i^2 A(\mathbf{x}_i)^\top A(\mathbf{x}_i) & \sum_i d_i A(\mathbf{x}_i)^\top B(\mathbf{x}_i) \\ \sum_i d_i B(\mathbf{x}_i)^\top A(\mathbf{x}_i) & \sum_i B(\mathbf{x}_i)^\top B(\mathbf{x}_i) \end{pmatrix}. \tag{7.12}$$

Note that the condition number of $\mathbf{S}$ only depends on the distribution of features in 3D space and not on the motion. This makes the sources of motion errors easier to understand. The accuracy of the results depend on the magnitudes of the optical flow projected onto each flow component respectively, while the ability to separate components depends on the distribution of features in 3D space. This means that the system does not collapse if translations are small, even if the translational direction is erroneously determined.

A number of experiments were performed, with simulated data generated as in Section 6.4. The true rotation was set to 2° and a translational magnitude of 6 cm was used. A priori depth estimates were perturbed using Equation 7.6. Results for translations in three different directions, forward, diagonally and laterally, are summarized in the table of Figure 7.2. The definition of bias and stability may be found in Tian et al. (1996).

	ω bias	ω stab	$\mathbf{t}$ bias	$\mathbf{t}$ stab	$\lvert\mathbf{t}\rvert$ mean	$\lvert\mathbf{t}\rvert$ std
Forward	0.006°	0.13°	1.32°	29.1°	1.90 cm	0.49 cm
Diagonal	0.004°	0.10°	29.2°	8.5°	4.47 cm	0.40 cm
Lateral	0.003°	0.06°	0.17°	2.9°	6.02 cm	0.22 cm

Figure 7.2. Rotational and translational bias and stability for translations in forward, diagonal and lateral directions and rotations of 2° per update.

The rotational results are indeed satisfactory compared to the results presented in Chapter 6, that were not based on a priori depths. However, the estimated translations suffer from a serious bias in direction, as well as magnitude. The reason can be understood as follows. Due to errors in image positions, a term based on the error variance will be added to the t_z components of $\mathbf{S}$. This will lead to an underestimation of this component. However, flow generated by forward motion is rarely confused with other components, which means that they will remain approximately the same. Thus when moving along the optical axis, the magnitude will be underestimated, but the direction will not change. The large directional bias diagonally is a result of this underestimation in combination with lateral translations. Translations along the x-axis are left unaffected, since in this direction t_z is already zero.

Fortunately, it is possible to improve the translational estimates. Since the rotation estimate is relatively reliable, the rotational component can be subtracted from the optical flow. The resulting flow depends linearly on the inverted depths. Thus a much simpler problem can be derived observing that

$$\mathbf{u}_{\mathbf{t}d_i} = \frac{1}{d_i}\left(\mathbf{u}_i - B(\mathbf{x}_i)\,\omega\right) = \begin{pmatrix} t_x \\ t_y \end{pmatrix} - t_z \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{7.13}$$

If t_z is given, a component due to forward motion may be added to $\mathbf{u}_{\mathbf{t}d_i}$, which should result in two constants, t_x and t_y. This means that optimization can be performed on t_z using the variances of these parameters. In fact, in most applications the range of possible t_z is known in advance. In our system the minimum is found using steepest descent, starting from an initial value within this range.

The optimization function based on variances is usually well behaved and contains only one extremum. Results from experiments using such an additional optimization can be found in Figure 7.3.

	t bias	**t** stab	|**t**| mean	|**t**| std
Forward	0.37°	8.2°	5.99 cm	0.24 cm
Diagonal	0.39°	5.8°	6.01 cm	0.32 cm
Lateral	0.17°	3.2°	6.02 cm	0.23 cm

Figure 7.3. Translational bias and stability for translations in forward, diagonal and lateral directions after additional optimization.

Experiments

A series of experiments was performed in order to determine how much a priori depths estimates are able to improve the performance of the method of Zhang & Tomasi (1999). The iterative approach of Section 6.5.2 was tested on simulated data using initial estimates based on the method given above. In order to evaluate the performance of the iterative process, the direct method for obtaining initial estimates was also tested in isolation. The third approach that was tested is the one presented in Section 7.2, where the iterative method was modified so as to take advantage of a priori depths. The value K, in Equation 7.7, was chosen such that the influence of the a priori depths was equivalent to about 10% of the total error.

In the simulations all translations lie on the plane defined by the x-axis and the optical axis, with rotations around the y-axis. It is well known that rotational components around other axes are relatively easy to find (Maybank 1987, Jepson & Heeger 1990). This means that in order to get an understanding of the overall performance, we only have to test a limited set of motions, where the confusion between translations and rotations is maximized. Figure 7.4 show the stability of translations in three different direction; along the optical axis, diagonally and along the x-axis. Similarly the rotational stability can be seen in Figure 7.5. Each group of bars represents different combinations of translational magnitudes and rotations.

Studying the results, what is most striking is that the iterative process is only able to significantly improve results in the forward direction and only if the translations are large enough. The problem of estimating depth from stereo is similar to the case of lateral motion. Since a baseline is typically larger than the translational magnitude, no improvements are seen for motions along the x-axis. In that case it is better to rely on the stereo estimates, than trying to further improve the results using translations that are too small. In the forward direction, motion and stereo estimates may instead cooperate leading to more reliable results. However, this is only possible if the translations are large enough. The inclusion of additional free variables to an error sensitive problem such as

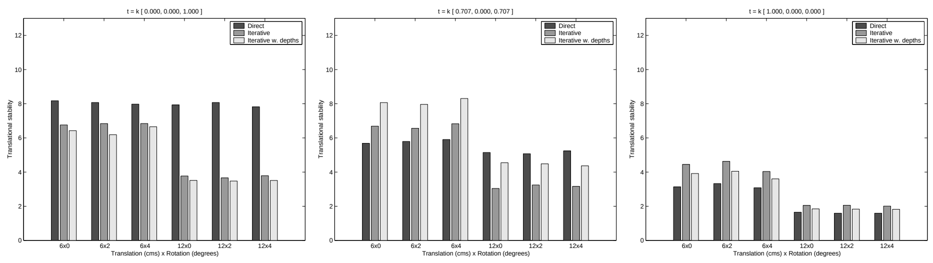

Figure 7.4. Translational stability for motions along the optical axis (left), diagonally (middle) and along the x-axis (right) for different combinations of true translation and rotation.

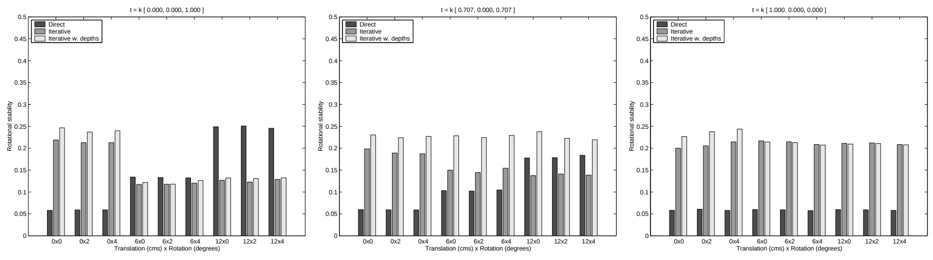

Figure 7.5. Rotational stability for motions along the optical axis (left), diagonally (middle) and along the x-axis (right) for different combinations of true translation and rotation.

this may thus lead to worse results, than if those variables are fixed to erroneous values.

One may summarize the results as follows. The use of a priori depths in the error function only seems to add to the complexity and rarely leads to more stable results. One possible interpretation is that an erroneous image position may well result in errors in depths, but that does not necessarily mean that the motion estimates are affected. The a priori depths originate from a different set of image pairs and should not be combined with features tracked in time. If the performance is measured in terms of motion, image errors along the translational flow direction will be disregarded, since such errors will be compensated by a sufficient change in estimated depth. However, if an a priori depth estimate is given, this freedom is lost. In order to adjust for errors in depth, the motion estimate will change, which means that additional motion errors will be introduced.

7.3 Ego-motion from 3D features

In the previous section, a priori depths were introduced in solving the structure-from-motion problem. Due to the dependency between image feature noise and triangulated depths, the benefit of the a priori depths was limited. An alternative approach might be one that does not rely on features in image space, but rather on features that have already been triangulated into 3D space. If image features are tracked in stereo as well as in time, it should be possible to derive the observer motion directly from such 3D features. In this section it will be shown how this can be achieved. Unlike the previously described methods, great care is taken in order to make the proposed system work in practice, that is robust statistics are used to eliminate outliers that arise from false matches or independent motion.

Assume that we have two sets of triangulated 3D features from two different instances in time. In the coordinate frames of the two camera positions, let the position of the *i*th feature be denoted by $\mathbf{x}_i$ and $\mathbf{y}_i$ respectively. If the set of features is rigid, the 3D positions are related through the rigid motion equation, that is $\mathbf{y}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t}$. The vector $\mathbf{t}$ represents the camera translation and $\mathbf{R}$ is an 3×3 rotation matrix. The ego-motion can thus be determined by minimizing the least square errors

$$f(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{N} |\mathbf{y}_i - (\mathbf{R}\mathbf{x}_i + \mathbf{t})|^2. \tag{7.14}$$

7.3.1 Estimating the rotation

A separation of $\mathbf{R}$ and $\mathbf{t}$ is possible observing that the two feature sets represent two similarly shaped clouds of points in 3D space. This translation will then be given directly by the difference between their centroids, that is $\mathbf{t} = \overline{\mathbf{y}} - \overline{\mathbf{x}}$. However, the existence of outliers will corrupt this estimate. Thus a few modifications will be introduced later on.

Subtracting the centroids of the clouds from the 3D points results in two new sets of positions, $\acute{\mathbf{x}}_i = \mathbf{x}_i - \overline{\mathbf{x}}$ and $\acute{\mathbf{y}}_i = \mathbf{y}_i - \overline{\mathbf{y}}$, and an objective function

$$f_1(\mathbf{R}) = \sum_{i=1}^{N} |\acute{\mathbf{y}}_i - \mathbf{R}\acute{\mathbf{x}}_i|^2. \tag{7.15}$$

Horn has proposed two alternative methods for solving for the rotation (Horn 1987*b*, Horn 1987*a*), either using quaternions or by fitting orthogonal matrices. However, in this study an elegant method of Arun et al. (1987) based on singular value decomposition (SVD) will be used instead. If the error function in Equation 7.16 is rewritten as

$$f_1(\mathbf{R}) = \sum_{i=1}^{N} (|\acute{\mathbf{y}}_i|^2 + |\acute{\mathbf{x}}_i|^2 - 2\,\acute{\mathbf{x}}_i^\top \mathbf{R}\acute{\mathbf{y}}_i), \tag{7.16}$$

it can be seen that the problem may be restated as that of maximizing

$$f_2(\mathbf{R}) = \sum_{i=1}^{N} \acute{\mathbf{x}}_i^\top \mathbf{R}\acute{\mathbf{y}}_i = \mathrm{Trace}(\mathbf{R}\mathbf{H}), \quad \text{where} \tag{7.17}$$

$$\mathbf{H} = \sum_{i=1}^{N} \acute{\mathbf{y}}_i \acute{\mathbf{x}}_i^\top. \tag{7.18}$$

Arun et al. (1987) showed that the rotational estimate $\hat{\mathbf{R}}$ that maximizes $f_2(\mathbf{R})$ can be found using a singular value decomposition. The correlation matrix $\mathbf{H}$ is factorized into $\mathbf{UDV}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are two orthogonal matrices and $\mathbf{D}$ is diagonal. Depending on its determinant, $\hat{\mathbf{R}} = \mathbf{VU}^\top$ will represent either a rotation or a reflection. A rotation is characterized by a determinant equal to $+1$, while -1 instead means that $\hat{\mathbf{R}}$ is a reflection. However, as long as all features are not located on the same plane, this is no serious problems, since then the determinant will always be $+1$.

Improving the results

The errors of the reconstructed features may unfortunately be very large, as described in Section 7.1. To improve the stability of the estimation process, features of large errors should be identified and discarded as early as possible. These features were hard to find in the method just described, since they are all hidden within $\mathbf{H}$. In the system presented here, the three dimensional displacement vectors $\mathbf{d}_i = \acute{\mathbf{y}}_i - \acute{\mathbf{x}}_i$ is instead analyzed prior to calculating $\mathbf{H}$. First of all, some erroneous features can be discarded because $|\mathbf{d}_i|/|\acute{\mathbf{x}}_i|$ is too large. For these points the corresponding rotation would simply be larger that what can be considered as likely.

One observation that can be exploited is the fact that all $\mathbf{d}_i$ should lie on the same two-dimensional plane. The normal of this plane is the rotation axis, which can be estimated as the least eigenvector $\mathbf{n}$ of

$$\mathbf{H}_2 = \sum_{i=1}^{N} \mathbf{d}_i \mathbf{d}_i^\top. \tag{7.19}$$

Once $\mathbf{n}$ is determined the expected rotational displacement direction is given by $\mathbf{r}_i = (\acute{\mathbf{x}}_\mathbf{i} \times \mathbf{n})/|\acute{\mathbf{x}}_\mathbf{i} \times \mathbf{n}|$ and an angle can be estimated as

$$\alpha = \frac{\sum_i \mathbf{d}_i^\top \mathbf{r}_i}{\sum_i |\acute{\mathbf{x}}_\mathbf{i}|}. \tag{7.20}$$

Simply using $\alpha\mathbf{n}$ as the estimated rotation is possible, but it is not as accurate as estimates based on $\mathbf{H}$. The reason for calculating α is instead that serious outliers can be found and eliminated using errors defined by $\mathbf{e}_i = \mathbf{d}_i^\top \mathbf{r}_i - \alpha|\acute{\mathbf{x}}_\mathbf{i}|$. It turns out that rotational estimates based on $\mathbf{H}_2$ are considerably less sensitive to large errors than if $\mathbf{H}$ were used directly.

If the SVD based estimates $\hat{\mathbf{R}}$ are studied in noisy situations, it is revealed that orientations are relatively easy to find, but the magnitudes are typically underestimated. As a consequence of that an additional nonlinear optimization on the magnitudes can be performed using Equation 7.16. Summarized, rotations are estimated using a three step procedure. First the rotational axis is estimated as the least eigenvector of $\mathbf{H}_2$ and through projections onto the expected rotational displacement vectors an angle is calculated. Features with errors larger than a given threshold can then be treated as outliers. Using the remaining features, a new rotational estimate is found using a singular value decomposition of $\mathbf{H}$. The last step includes an nonlinear optimization on the rotational magnitudes directly using the error function in Equation 7.16.

7.3.2 Translation estimation

It was previously mentioned that a translation $\mathbf{t}$ can be estimated using the difference between the centroids of the two clouds of points. This is done in the first step out the presented algorithm. Prior to that, gross errors are avoided as follows. Since rotations around the camera centre do not change the individual feature depths and since the translational speed is expected to be limited, 3D features with too large differences in depth between two consecutive frames are likely results of mismatches. In the current implementation features are removed, if the difference between two frames is more than the sum of 30 cm and 10% of the first depth, or if either depth estimate is negative. Fortunately, mismatches typically result in considerable differences in depth and the vast majority of erroneous features due to mismatches can thus be identified. In practice, this is the most important advantage of using triangulated features from two different instances in time.

In order to update the translation, the current rotational estimate $\hat{\mathbf{R}}$ is used for the stabilization of features projected onto the image plane of the left camera. The translation is then determined in 2D from the optical flow using least squares. Since depths are available, this process is relatively stable. The reason for not performing the operation on the 3D features, but rather on their projections, is the uncertainty in position along the optical axis. The problem could have been solved using a covariance matrix to weight feature data, but this would in fact have led to a problem similar to the 2D case. Solutions in 3D tend to vary depending on the true translational direction, with errors large in either direction or speed. However, the mean errors do not differ much from those of the 2D case.

The overall motion estimation process is executed iteratively. Rotations and translations are determined in sequence, updated using previous motion estimates. From the second pass on, rotations are estimated using the current estimate of $\mathbf{t}$, rather than the difference between cloud centroids. In the current implementation eight passes are used in total, even if convergence is typically reached within three to five passes. Within each pass outliers are reidentified, using the complete set of features. Based on the current motion estimate, features with errors in projected image positions larger than a certain threshold are removed from the next pass. If this process were instead based on 3D positions, features located far from the observer might be eliminated, even if they were correctly matched between frames. This leads to a poor balance of features at different depths. In order to limit the effect of erroneous 3D positions, an additional outlier identification is performed before calculating the rotation $\hat{\mathbf{R}}$, as was described above.

7.3.3 Experiments

The complete ego-motion estimation process was evaluated using simulated data as in previous experiments, with one exception however. Instead of just using 80 randomly generated 3D features, an additional 20 non-rigid points were added, so as to test the behaviour when independent motion is present. These outliers originated from a rotating cylinder located at the centre of the scene. Its rotational axis was along the y-axis and the angular speed $10°$ per frame. While rotating the cylinder was also translating along the same axis, at a speed of 10 cm per update. Since erroneous features that exist due to mismatches are easily found in the outlier detection process, such features were not added. The remaining 80 features were evenly spread within the truncated pyramid between 200 and 600 cm from the observer.

All 100 features were then projected onto 384×288 pixel image planes of the left and right cameras, with 0.7 pixel in standard deviation noise added to each coordinate. With image features from two consecutive instances in time and a baseline of 20 cm, two sets of 3D features were created using triangulation. A systematic error of $0.3°$ was further added to the vergence angles, in order to simulate the effect of incorrect camera calibration. To adjust for such errors, the shapes and sizes of the two point clouds are analyzed early on in the process. The adjustment is done adding a constant to disparities in one of the two image frames, such as to make shapes and sizes approximately the same.

Results

In Figure 7.6 results from a series of simulations have been summarized. The rows each represent the results from 1000 test runs, for different choices of true translational direction $\angle\mathbf{t}$, magnitude $|\mathbf{t}|$ and rotational speed $\mathbf{R_y}$. All rotations are given in degrees around the y-axis, with translations in the plane defined by the optical axis and x-axis. The translational directions are expressed in degrees from the optical axis and the magnitudes in centimetres per update. The last three columns show the variances of the same parameters being estimated.

Unlike all previously described ego-motion estimation methods, that were based on optical flow, the presented approach turns out to be invariant not only to the rotational speed, but also to translational direction. Furthermore, no significant bias was noted in any of the parameters. In previous iterative methods, large errors in rotation were typically accompanied by large errors in translation. In cases of small translations and low rotational speed, the second order factors of the rotational flow were easily drowned in image noise, resulting in corresponding errors in translation. These errors were especially evident in the forward direction, since here the required angular change in translation is

$\angle\mathbf{t}$	$\lvert\mathbf{t}\rvert$	$\mathbf{R_y}$	$\angle\hat{\mathbf{t}}$	$\lvert\hat{\mathbf{t}}\rvert$	$\hat{\mathbf{R}}_\mathbf{y}$
	0	0		0.43	0.039
	0	2		0.42	0.039
	0	4		0.42	0.037
0	3	0	4.31	0.40	0.038
0	3	2	4.25	0.40	0.038
0	3	4	4.76	0.43	0.040
0	6	0	2.32	0.40	0.037
0	6	2	2.33	0.40	0.039
0	6	4	2.33	0.43	0.039
45	3	0	4.56	0.42	0.040
45	3	2	4.38	0.41	0.039
45	3	4	4.68	0.44	0.042
45	6	0	2.32	0.45	0.040
45	6	2	2.24	0.43	0.040
45	6	4	2.38	0.45	0.041
90	3	0	4.25	0.40	0.037
90	3	2	4.22	0.43	0.040
90	3	4	4.66	0.45	0.043
90	6	0	2.19	0.43	0.043
90	6	2	2.25	0.43	0.042
90	6	4	2.23	0.46	0.045

Figure 7.6. The standard deviation of the estimated translational direction and speed, and pan rotational speed shown as functions of their true values.

maximized, in order to compensate for a given rotational error. In the stereo case, however, there are no such limitations and the rotation may be recovered using the relatively accurate depths estimated from stereo. Thus the required translational compensation is much smaller.

If the translational direction errors are analyzed for a broader range of true magnitudes, it can be seen that the product of directional errors and magnitudes are approximately constant. There is one possible explanation to that. Assume that a translational vector of length a is given and the error perpendicular to the vector is x, then the angular error is $\theta = \tan(x/a) \approx x/a$ for large enough translations. This means that $\theta a \approx x$, which can be assumed to have a constant variance, since the baseline has a fixed length.

7.4 Independent motion detection

Once the motion of the observer has been determined and the three dimensional structure of the scene is available, it is possible to find image regions of independent motion. In a rigid scene only the ego-motion is required for the optical flow to be predicted, assuming that the depths are already known. If this optical flow field is different from the one estimated as in Chapter 2, the corresponding parts of the scene cannot be rigid, that is they belong to independently moving objects. Unfortunately, the optical flow calculated in Chapter 2 is not accurate enough for such a direct comparison. However, since the optical flow determines the displacement from one image frame to the next, not only the optical flow can be predicted, but also the image data itself. Thus image data may be warped from one frame to the following and a comparison can be done between images, not between optical flow fields.

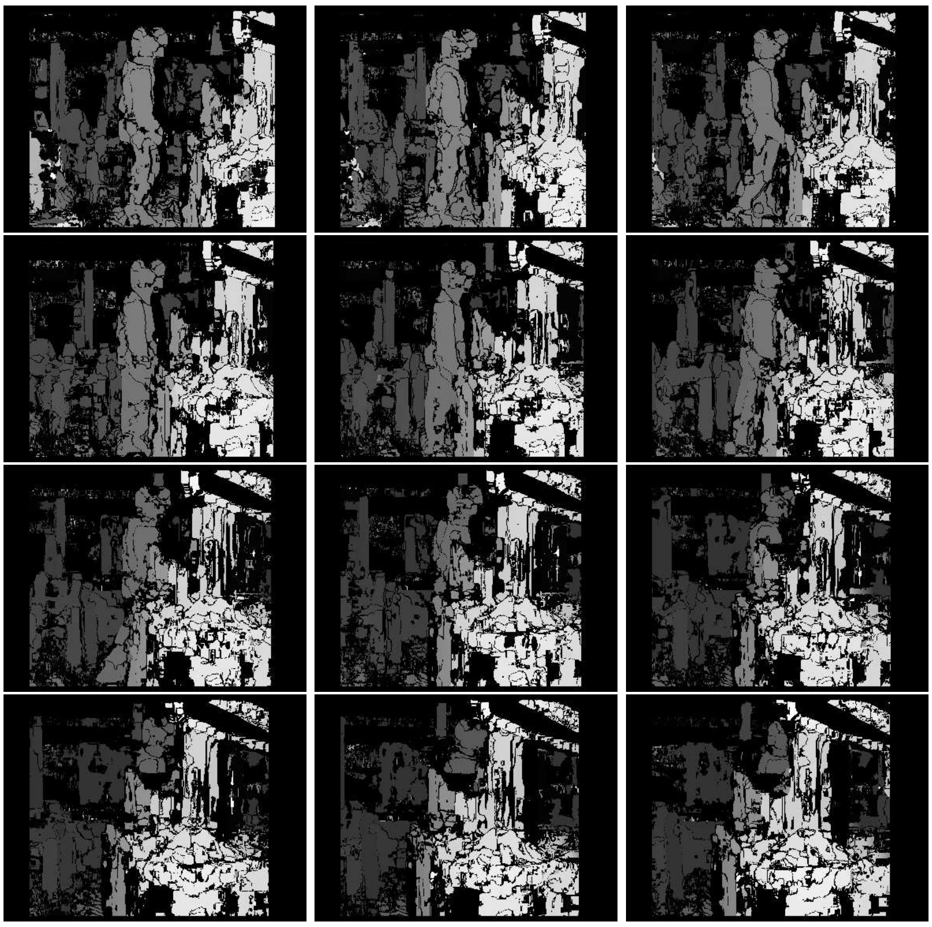

Figure 7.7. Disparities calculated using area correlation and sums of squared differences. Black areas indicate pixels for which no disparity could be found.

First assume that disparities have been calculated using one of the methods described and evaluated in Section 5. Figure 7.7 shows a sequence of disparity maps determined using the simplest of these methods, based on area correlations and sums of absolute differences. The number of false positives have been reduced, matching the left and right images and both directions. No additional postprocessing has been performed. It should be emphasized that the camera calibration was done automatically using the iterative method proposed in Chapter 4. Given the disparities, depths can then be calculated as

$$Z_i = b\frac{f}{d_i}, \tag{7.21}$$

where d_i is the disparity at image position $\mathbf{x}_i$, b denoting the length of the baseline and f the focal length expressed in terms of image pixels.

In the previous sections of this chapter, randomly generated sets of feature points from two sequential images were used to estimate the motion parameters. In practice these features have to be determined using some kind of feature detector. For the experiments presented here, the Harris corner detector (Harris & Stephens 1988) was used. Matching is then performed between the two images, in both directions, using modified normalized cross-correlation of 9×9 pixel areas. In fact, matching is done in motion as well as in stereo, since the reconstructed depths are used in the ego-motion estimation process. The stereo matching is done similarly prior to the camera calibration. This is both a strength and a weakness. Since a corner feature has to be visible in four different images from two separate image pairs, sporadic mismatches tend not to survive and are thus excluded. However, the same applies to less distinct corners, which might result in too few features being left for ego-motion estimation.

In Figure 7.8 an example of extracted corner features can be seen, with lines illustrating their corresponding matches in time. Using the method presented in Section 7.3, the majority of outliers, which are shown in white, belong to the person walking in the centre of the images. Unfortunately, some correct features located far from the observer have also been eliminated. This is likely to be a consequence of the temporary outlier detection performed just before rotations are estimated. There is a tendency towards nearby features dominating the ego-motion estimation process, since errors in depth are smaller for these features. It should be kept in mind that the threshold used for the exclusion of outliers was set to approximately one pixel, which is relatively low.

Once the motion parameters have been calculated, that is when the translation $\mathbf{t}$ and rotation ω are known, the predicted optical flow is given by

$$\mathbf{u}_i = Z_i^{-1} A(\mathbf{x}_i)\,\mathbf{t} + B(\mathbf{x}_i)\,\omega. \tag{7.22}$$

A comparison between the first image and a warped version of the following one can then be performed. The residual

$$e_i = |I_{t+1}(\mathbf{x}_i + \mathbf{u}_i) - I_t(\mathbf{x_i})| \tag{7.23}$$

may then be used to determine areas, where the rigidity assumption does not hold. The ordering of the two frames does not really matter, but since the positions $\mathbf{x}_i$ are defined in the first image, the second one is warped. In practice, this

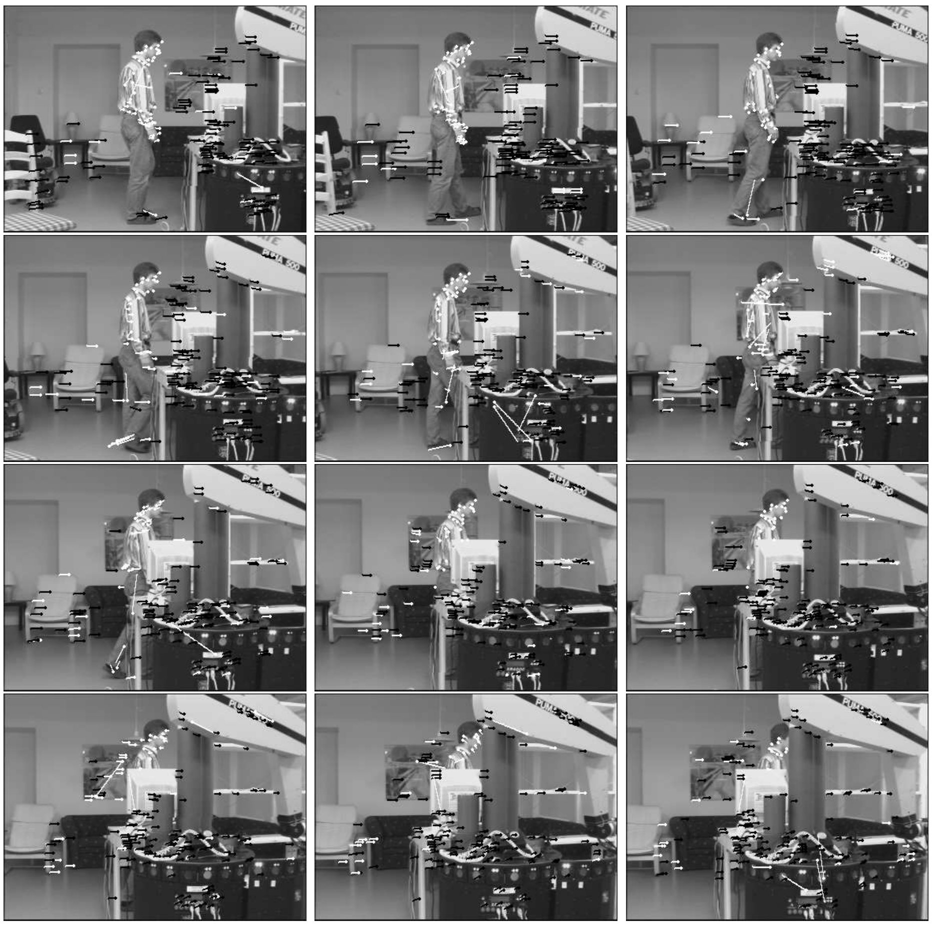

Figure 7.8. Corner features used for estimating ego-motion, with lines indicating corresponding matches between image frames. Outliers are shown in white.

ordering is preferably reversed, so that the residual corresponds to independent motion regions of in the most recently grabbed image.

The data in Figure 7.9 show pixels for which the residual e_i exceeds 25 in pixel value. For areas where there are no disparity estimates, that is the black areas of Figure 7.7, the mean disparity was used instead. These areas exist either due to the lack of image structure or because the left-right consistency check fails. The most notable such areas are visible to the far left of the images or near the manipulator arm in the front. There is also a large number of sporadic errors, but these are typically only about one pixel in width, which indicates that they originate from pixel noise around image gradients. The remaining residuals exist near the walking person in the centre of images. For these regions of independent motion, the optical flow is different than what was predicted, resulting in large errors in the subtraction.

The results could probably be used as they are, with a blob detection algorithm working directly on the residuals. However, a better segmentation may be obtained if disparities are also taken into consideration. After all, a visible object moving in an environment occupies not just a certain part of the images, but also of 3D space. In the algorithm presented here, the following operations are performed. First a two-dimensional histogram is created, with one dimension representing x-wise positions and the other one disparities. Each cell in the histogram is determined by the number of pixels with optical flow and disparity values matching the particular cell and a residual larger than 25 pixel values. Pixels with unknown disparity values are disregarded.

From the histogram, peaks are extracted, each representing a region of independent motion or noise. The extension of such a region is determined fitting a quadric surface around the peak and measuring its curvature. The y-wise extension is calculated from the residuals, using pixels located close to the histogram peak and within its extension. Thus two different moving objects cannot be located in the same position both in x and in depth. This does not matter, however, since that would imply that the objects are placed on top of each other in the 3D world, which is not likely to happen. Once the extension and position of the moving region are known, segmentation can be performed using the disparity map. After segmentation each pixel is hypothesized as being either in the foreground or background.

The light areas of Figure 7.10 show results obtained using this approach. An additional morphological operation was applied to the segmented data, such as to fill in empty regions, where low contrast did not permit disparities to be calculated. The operation was implemented as follows. For each pixel the number of foreground pixels within a four pixel radius is counted, using a circular filter swept over the segmentation data. For low contrast pixels its final state, foreground or background, is determined by the majority of pixels agreeing on the same state. The same operation is applied to high gradient points, but in this case at least 80% of the nearby pixels need to be of the opposing state in order for the state to be changed. Through this approach data will be spread more in uniformly shaded regions than in regions rich in texture.

Postprocessing was done at the final stage of the algorithm and not on the original disparity map. This is because there are only two possible hypotheses in this case. If there were instead multiple hypotheses, especially when there is no natural metric, postprocessing might lead to corruption of the data. With only two hypotheses, this corruption is likely to be smaller. Furthermore, if corruption of data cannot be avoided, postprocessing should preferably be performed as late as possible, so as not to affect the performance of following procedures.

It should be emphasized that no temporal consistency was taken into consideration in the results presented here. Each blob shown in Figure 7.10 is calculated based on information from only two instances in time. Without any integration of results, the blob disappears as soon as the object stops moving. It is likely that data from multiple frames would improve results. In fact, the results are so satisfactory, that they may be used directly to track the object in question. On the other hand, in order to track the object, as much information as possible should be extracted from the blob, improving speed as well as accuracy. The computation should preferably be performed locally and not on the whole image. However, a framework is created in which tracking can be initialized.

7.5 Conclusions

In this chapter three-dimensional feature points were exploited for ego-motion estimation. Using matches between image features in stereo, the corresponding depths were determined using triangulation. These points together with the bilinear optical flow constraint could then be used to find the translation and rotation of the observing platform. However, if the method of Zhang & Tomasi (1999) described in Chapter 6 was initialized and executed using these motion estimates, the results could hardly be improved. It seems as if depth estimates based on stereo and motion compete, rather than cooperate. This is probably because depths were determined using different pairs of noisy image feature pairs, which leads to different depths being estimated, even if both stereo and motion pairs ought to result in the same reconstructed depths in a noise-free case.

This led us to try another scheme which did not involve the optical flow constraint. Instead the motion parameters were estimated directly from 3D features from two different instances in time. The method was implemented with outlier detection so as to make it robust in a real situation. A direct quantitative comparison between this method and the monocular ones is not fair, but it is worth noting that the robustness was significantly improved using stereo, even when 20% outliers were added to the data set. The quality of results seemed to be invariant not only to rotation, but also to the translational direction, which is very different from the monocular case.

In the final part of the chapter we showed how regions of independent motion may be found using disparities in combination with the ego-motion. The results suggest that they may be used for tracking and not just for the initialization of tracking. However, once enough information has been extracted from moving regions, one could use a simpler method, that does not work on the whole image. The current implementation of the complete algorithm requires about 90 ms in computational cost on a 1.2 GHz Athlon, which ought to be sufficient for the redirection of gaze in a complete active system. The independent motion information will later be integrated with various other cues in Chapter 8.

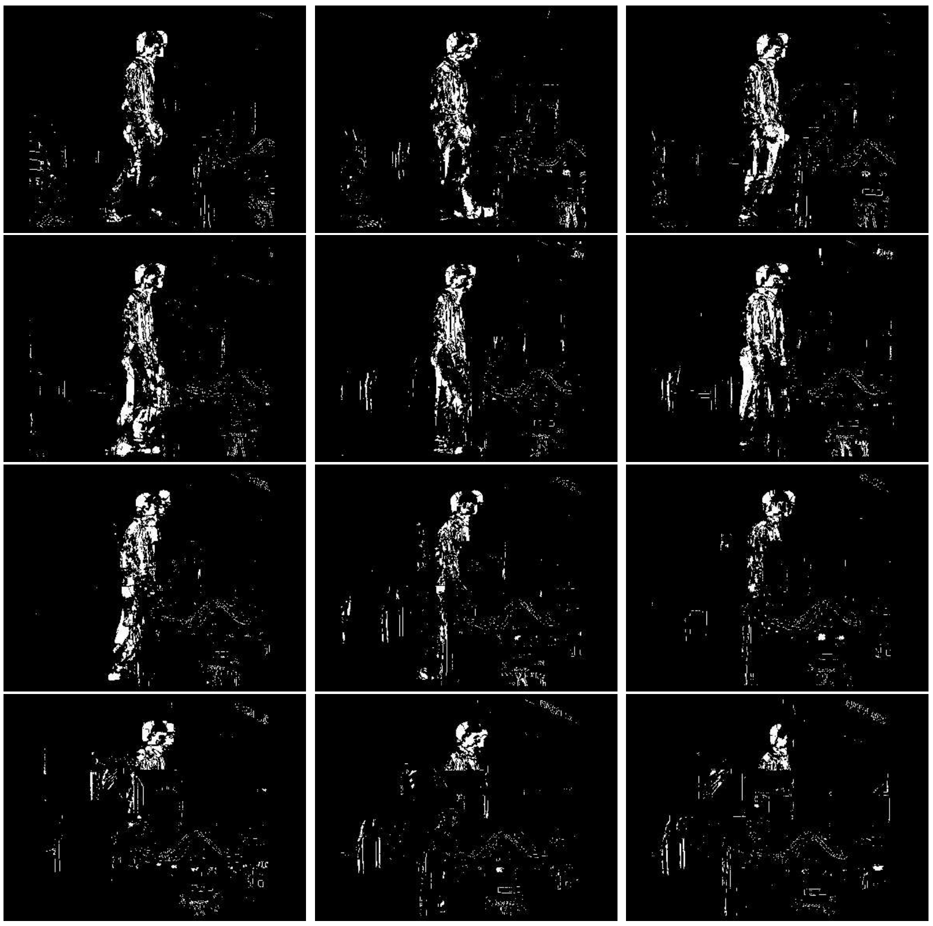

Figure 7.9. Residuals after subtracting a warped version of a prior image from the following true one. White areas either come from independent motion or as a result of disparity and image pixel errors.

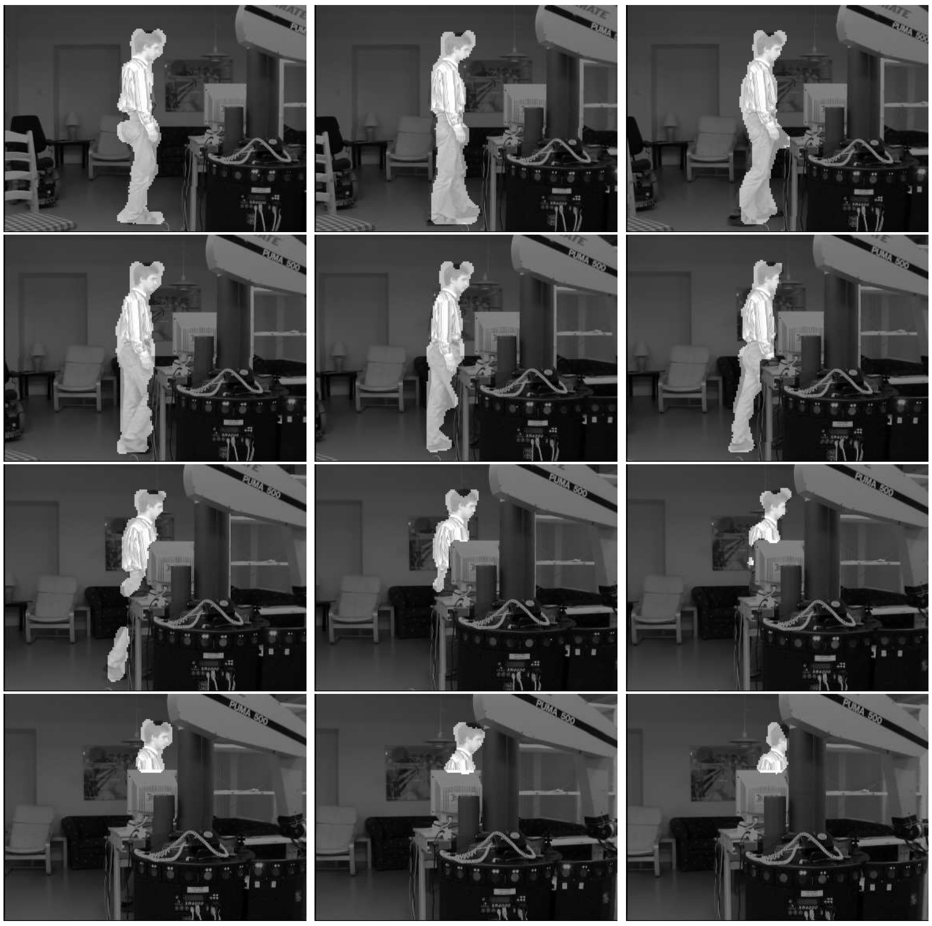

Figure 7.10. Regions of independent motion.

Chapter 8

System integration

In this chapter the components explained and analyzed in the preceding chapters will be integrated into a complete system. The system is intended to be used to control the direction of gaze of a vision-guided robot. Attention is paid to the implementation details that were not made clear earlier on. The intention is to give an overview of the components used and the flow of information from visual input to an updated gaze direction. The system contains three different feature paths, with results respectively in terms of disparities, optical flow and regions of independent motion. It will be shown how these cues may be integrated into a common representation and how top-down control could be included. Since stereo and motion data typically exist only around high gradient edges, data will often be missing within textureless areas. Towards the end of the chapter a method based on graph cuts will be given that may be used to fill in such missing information.

8.1 Flow of information

In the diagram of Figure 8.1 the operations and cues involved in the system have been summarized. The boxes indicate resulting data, whereas rounded boxes represent operations necessary to produce such data. Furthermore, the incoming images are supposed to be available within each individual operation. This is by no means a final system. Additional cues such as colour and texture cues, blobs and ridges should be provided in the future, as well as functions for storing and retrieving visual object data. Unfortunately, neither stereo nor motion cues can be used directly for that purpose. However, important information about size, shape and rigidity may be derived from these cues.

The flow of information is more or less bottom-up, in that every piece of information is derived directly from the visual input. The presented system is not strictly feed-forward, since adaptation and guidance may be possible through feedback links, either locally or between different components. Non-visual cues may also be be available, such as inertial information from the observing agent. A bottom-up system is able to act on unexpected events, but can hardly be used for performing any tasks involving interaction with the environment, since that would require that information about the object and the environment is stored, either short-term or long-term. Knowledge that is not directly available in the visual input, but remembered from previous experience or hardwired into the system, consistutes the top-down part of an envisaged system. Later on in Section 8.3, the problem of integrating different cues will be briefly considered, with a discussion on how to integrate top-down control into the system.

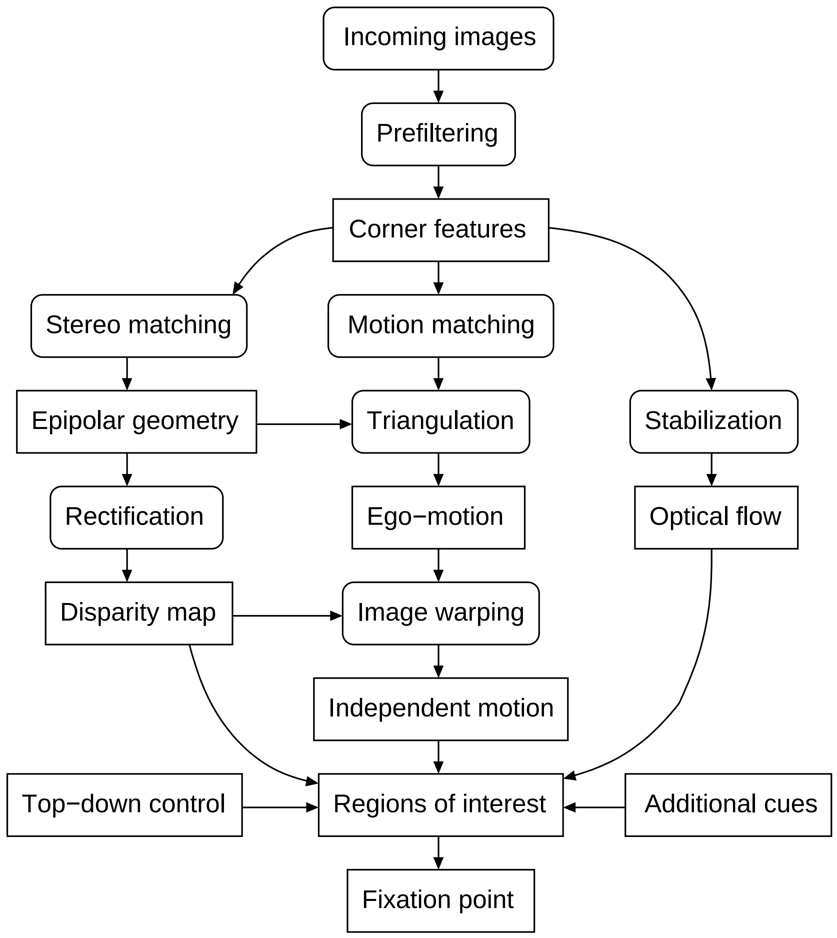

Figure 8.1. The flow of information with boxes representing data that have been calculated through operations shown as rounded boxes.

8.1.1 Feature paths

As can be seen in Figure 8.1 the system basically consists of three parallel feature paths, a disparity path, an optical flow path and an independent motion path. Each path results in a dedicated map, which will be used as a basis for determining an updated gaze direction.

Disparity The disparity path consists of four sequential steps, corner matching, epipolar geometry estimation, rectification and calculation of disparity maps. The matching of corners will be discussed in the following section. In choosing the proper epipolar geometry algorithm, one has to consider the environment in which the system is intended to work, as was discussed in Chapter 4. The essential matrix is preferable if one does not know anything about the depths of feature points and if the vergence angle is large. The increased complexity of nonlinear optimization is hardly justified in most situations. The speed of the iterative approach, described in Section 4.9, is attractive, but relies on the median depth of feature points in space being approximately known. However, this is often not a serious limitation. Just knowing the size of the intended environment has shown to be enough for this approach to work. Consequently, the system presented here uses a combination of the iterative approach and the linear method based on essential matrices. As long as the vergence angles stay within about 10°, the convergence rate of the iterative approach will be acceptable and the system keeps the results from this method. The calculated median depth from one update is used as an input in the next update. Upon failure the system switches to the linear approach and continues using it until the iterative approach can be resumed.

Rectification is a process in which the stereo images are projectively transformed, as if they were captured by two parallel cameras. Rectification is necessary, since most disparity algorithms rely on epipolar lines being parallel to the scan-lines, in order to run in real-time. The complexity of the algorithms can then be greatly reduced and a parallel implementation is easier. The projective transformation involves a bilinear filtering of the images. Since such a filtering might influence the correlations done when disparities are calculated, it is preferable to use similar filtering for both images. If the vergence angles are relatively small, it would be possible only to transform one of the two images, reducing the computational cost of the rectification. However, the best results are achieved if both images are transformed, using approximately the same rotation around the y-axes for both images. A benefit of the geometry estimation process is that one gets a good idea of the range of available disparities. During the rectification images can thus be shifted, so that the range of disparities matches that of the disparity algorithm.

For binocular disparities the simplest possible method was chosen, that is a method based on area correlation with sums of absolute differences. In Chapter 5 it was shown that the major difference between algorithms is primarily the resulting density and not the accuracy itself. More complex algorithms can fill in missing information, between regions rich on texture, where disparities can be found locally. For a complete "seeing" system it is worth considering where, in the flow of information, this filling-in operation should be performed. Typically objects at different depths are attended to separately and not at the same time. Hence, filling-in could be performed later in the path, using information from additional cues, and not necessarily within the disparity algorithm itself. How this can be done in practice will be shown later on in Section 8.4.

Independent motion In the independent motion path, the objective is to find image regions, where the optical flow does not match that of a rigid scene. In theory it would be possible to get the same results from the optical flow path in conjunction with disparities, but the data are unfortunately not good enough. Optical flow is still an important cue and will be calculated in parallel. The first step of the path is the motion matching, which was described earlier. Only features that have earlier been successfully matched in stereo will be taken into account. Thus the number of feature pairs will be somewhat reduced. The following ego-motion procedure, that was described in Section 7.3, is however not as sensitive as the epipolar geometry process. After features have been matched they are triangulated based on stereo, such that their positions in 3D will be made available. Three-dimensional points from two consecutive image frames are then used for ego-motion estimation.

The triangulation is performed as described in Section 7.1, but with the vergence angle and gaze direction from the estimated epipolar geometry, rather than the approximation of Equation 7.2. With $\mathbf{R}$ and $\mathbf{s}$ denoting the rotation and translation as defined in Section 4.3.1, the depths are found by solving the following problem with least squares,

$$\begin{pmatrix} |\mathbf{x_l}|^2 & -\mathbf{x_l}^\top \mathbf{R} \mathbf{x_r} \\ -\mathbf{x_r}^\top \mathbf{R}^\top \mathbf{x_l} & |\mathbf{x_r}|^2 \end{pmatrix} \begin{pmatrix} z_l \\ z_r \end{pmatrix} = \begin{pmatrix} \mathbf{x_l}^\top \mathbf{s} \\ -\mathbf{x_r}^\top \mathbf{R}^\top \mathbf{s} \end{pmatrix}. \tag{8.1}$$

The homogeneous coordinates $\mathbf{x_l}$ and $\mathbf{x_r}$ are the feature positions in the left and right image planes and the corresponding 3D position is given by $\mathbf{x} = z_l \mathbf{x_l}$.

Once the ego-motion and disparities are known, a forward prediction of the next image frame can be made. The predicted image is then subtracted from the next image, so as to create a residual image which indicates areas where independent scene motion is most likely. For an active observer working in a real dynamic environment, this is a very strong cue, since it might indicate unpredictable events occurring in the scene.

Optical flow The optical flow path consists of two separate components, image stabilization and optical flow calculation. Stabilization is done using the method based on corner features, that was described in Section 3.2.1. If this method collapses, that is the number of matched pairs are too few, the other method based on image gradients is used instead. This typically happens when most parts of the visible scene are located nearby and variations in depth are small.

The matching of corners in two consecutive frames is done intrinsically within the algorithm, even if the corners matched in the motion matching operation could in fact be used here.

The reason for doing stabilization is that optical flow is only accurately calculated, if the magnitudes of the flow vectors are limited. The complexity of the optical flow methods increases radically if larger magnitudes are permitted. In the current implementation, the method of Lucas & Kanade in Section 2.2.1 is used because this was the only method that was fast enough, out of those tested in this study. As long as the magnitudes are limited the accuracy is reasonably good, even if the method has problems along discontinuities. In the future the stabilization should work in two different modes, depending on whether an object of interest is tracked or not. During tracking stabilization ought to be done on the object being tracked, rather than on the dominant background.

Regions of interest

The last component in the flow of information is a procedure that combines all available feature maps into a common representation, from which the next gaze direction can be found. This part has not yet been implemented into the system, since object specific cues such as from colour and texture still remain to be included. Since stereo and motion cues are not object specific, they cannot be directly used for object representations, that are to be stored for long-term use. The system should not just include bottom-up information in terms of incoming sensory data, but also top-down knowledge of previously encountered objects and events. Such knowledge is important for the observer to engage in tasks that require interaction. In Section 8.3 the problems of integrating cues and top-down knowledge into the system will be discussed and a common framework proposed.

8.2 Preprocessing and implementation

This section includes a description of the implementational details that are necessary for the system to work in practise. Before cues can be extracted using the feature paths presented earlier, the incoming images have to be preprocessed, so as to limit the effects of radial distortion and image noise.

Preprocessing

In the presented system the radial distortion is compensated for through a warping procedure. Due to distortions in the lenses objects appear to be rounder than they are in reality, which explains why it is often known as "barrelling distortion". The further away a pixel $\mathbf{x} = (x, y)^\top$ is located from the optical centre $\mathbf{c} = (x_c, y_c)^\top$, the more the image has to be compressed. Most radial distortion models are thus based on the the image distance $d = \parallel \mathbf{x} - \mathbf{c} \parallel$. For the presented system the following simplified model was applied,

$$\mathbf{x}' = (1 + \kappa d^2)(\mathbf{x} - \mathbf{c}) + \mathbf{c}. \tag{8.2}$$

Here $\mathbf{x}'$ denotes pixel positions after compensation and κ is a scale factor that determines the amount of compensation necessary. Higher order terms of the scale function were disregarded, since it was experimentally concluded that these only had a small effect on the outcome.

For the cameras and lenses used during the work of this thesis, the factor κ was usually set to a value in the range between 0.04 and 0.06, assuming that all image positions have been normalized using the focal length. The values were found off-line, with an approach based on Canny's edge detector (Canny 1986). The method uses the assumption that the observed scene contains a large number of straight edges, which is typical in indoor environments. Locally straight edges are first extracted from a couple of images. For all edges of sufficient length, a two-dimensional curve is fitted to pixels along the edge. From each curve the end-points and centre are found. Since these three points ought to be located on the same line, the determinant of the matrix, which has the points in homogeneous coordinates as its columns, should be zero. An optimization function is then created as the sum of all determinants, with which the parameters κ, x_c and y_c are found iteratively using gradient descent.

When the radial distortion factors are known, an incoming image may be warped into a new image, for which the distortion is reduced. In the presented system this is done using a bilinear filter. Unfortunately, in addition to ordinary image noise, new errors will be introduced in this procedure. Thus images have to be low-pass filtered right after being warped. This is done using a small, separable linear filter. The size of the filter is kept as small as possible, since different algorithms may require different amounts of prefiltering and unnecessary filtering should be avoided, in order not to hurt the accuracy. Typically, operations such as corner extraction and disparity calculation perform additional low-pass filtering intrinsically.

Corner extraction

In the current implementation corners are extracted for three different purposes; for epipolar geometry estimation, stabilization and in order to determine the ego-motion. As mentioned previously in Chapters 4 and 7, corner features are found using the Harris corner detector (Harris & Stephens 1988). A second moment matrix is created using the gradients in a neighbourhood around each image point. From these matrices high curvature corner points are found. Gradients are calculated using two 5×5 pixel filters, one for each dimension. Unlike some extractors, such as SUSAN (Smith & Brady 1997), the Harris detector suffers from the fact that this filtering may affect the positioning of the corner points and move them in the direction of the centre of curvature. However, in this study the Harris detector is used anyway. One reason is that the operations involved can easily be parallelized, using hardware such as the MMX capabilities of recent Intel and AMD processors. Another reason is the fact that, even if corner positions are incorrect, the relative positions between two images are of greater importance and these will not change as much, since the errors are similar in the two images.

In practice the number of extracted features is of the order of 300 in a typical indoor scene. This number may be adjusted using a threshold on the curvature measure. In order to adapt to changes in illumination and structure, this threshold is updated dynamically, such that the number of corners will be kept as close to constant as possible. However, a predefined range of allowed thresholds constrains the updates, so that extreme values can be avoided. This is another reason why SUSAN is not used, even if this extractor may be used to find the more exact location of a corner in a second phase. With SUSAN the neighbourhood pixels, that differ in luminance by more than a certain value from the pixel in question, are counted. The extractor then operates on the number of such pixels. This means that there is no natural threshold with which the extraction can be controlled. As a consequence the number of corners extracted may vary considerably when the observed scene changes.

Corner matching

Corner features are matched in time as well as between stereo pairs. The goal is to find a sufficient number of correctly paired features that can be used for ego-motion and epipolar geometry estimation. A pair is correct if the matched features represent the same point in 3D space. That procedure typically breaks down if the feature pairs are too few. This is in fact the most critical weakness of the whole system and failure must be handled gracefully. The current implementation uses a combination of prior estimates and motor counter data, to cope with image frames for which the corners are few and ensure continuous operation. A future system could be based on not just corner features, but also high gradient edges.

Corners are matched using modified normalized cross-correlation of 9×9 pixel image patches located around each extracted corner. A feature in one image is matched towards all possible features in the other and for each combination a corresponding matching score is calculated. Based on matching scores a favourite in the opposite image is found for every corner feature. If no score is good enough, no favourite is selected for the particular feature. This procedure is performed in both directions, and if two features are each others' favourites they are considered as a valid feature pair and stored for further processing. That is, if $\mathbf{x_r}$ is the favourite of $\mathbf{x_l}$, then $\mathbf{x_l}$ must be the favourite of $\mathbf{x_r}$, in order for $(\mathbf{x_l}, \mathbf{x_r})$ to be accepted as a valid pair.

In order to avoid a combinatorial explosion when corner features are matched, care has been taking to minimize the number of required matches. Assuming an image size of 384×288 pixels, a motion match is searched within a 50×50 pixel window, which is centred around a point defined by previously calculated ego-motion. The window is slightly different for stereo matching. By constraining the possible camera configurations, matches are only searched within a region, similar to the one determined by Equation 4.40. In the horizontal direction, the region is limited so that a triangulation always will result in points in front of the cameras. Thanks to these windows, the number of matches performed in total is radically reduced and the computational cost much lower than would otherwise have been the case. In order to avoid unnecessary cache misses, corner patches are stored in a separate location, aligned in memory so that 8 pixels can be compared in parallel using MMX hardware.

The valid pairs typically represent about 40% of the available corner features. Additional pairs can be created, letting the same matching operation be performed once again on all features that have not yet been paired. Thus features may be paired with their second best favourites, which is acceptable as long as the matching score is good. In the end the valid matches often include slightly more than 50% of all features. Unfortunately, the valid pairs hardly increase in number, if multiple rematch rounds are used. Because of occlusions some features only exist in one of the two images and will never be able to be properly matched. The final set of valid pairs usually contains only a few false matches. Empirically, it can be seen that the vast majority, 80%-90%, of pairs considered as valid are in fact correct, in that they originate from the same point in 3D space. False matches usually exist due to repeatable patterns in the image. Since a threshold is applied on the matching score, occlusions rarely lead to mismatches. The picture is quite different in the case of wide baseline stereo, especially if there are large differences in scale and rotations around the optical axes (Pritchett & Zisserman 1998, Tell 2002).

8.2.1 Implementation

The presented system has so far been implemented on two different dual processor platforms. The first one is an SGI Octane machine with two 195 MHz R10K processors, and the other is a platform based on 1.2 GHz Athlon MP, running the Linux operating system. The platforms and processors are slightly different in their characteristics. The Octane has got a relatively fast memory system and typically executes floating-point operations faster than integer ones. Due to the much smaller data caches of the Athlon processors, they suffer more from poorly planned memory allocation. Suggestions on how to maximize speed through efficient memory use and organization can be found in Appendix A.

Everything was programmed in C++, which means that it was rather easy to port the code from one platform to the other. However, in order to improve the overall speed on the Athlon machine, disparities are calculated and corners extracted using the MMX instruction set, the parallel hardware of recent AMD and Intel processors. With cue integration being excluded, the implementations on R10K and Athlon processors run with update frequencies of about 6 Hz and

9 Hz respectively. Due to the hardware specific implementation, the Athlon version runs faster even if only one processor is used.

On Athlon processors, the computational cost of each feature path is about the same. The most costly path, the disparity path, requires about 30 ms in total, while the independent motion path only uses about 18 ms and the optical flow one 25 ms. The feature maps and images are 384×288 pixels in size, except for the optical flow path, which functions at half this resolution. Additional time is spent on frame-grabbing, corner extraction and prefiltering. This leads to a total cost of approximately 110 ms. Unfortunately, when executing multiple operations in sequence, the total cost is typically more than the sum of each individual component. Since cache memory is scarce, different components compete for the same limited memory resource. Thus one should be careful not to draw any rash conclusions from components studied in isolation.

8.3 Cue integration and attention

Our intention is to create a artifical "seeing" system consisting of both bottom-up and top-down processes, so that gaze can be directed not just based on familiar objects entering the scene, but also due to unexpected event that might occur. We have mainly been concentrating on stereo and motion cues, since these cue are important in order to determine where objects are located in 3D space. However, a working "seeing" system would also require object specific cues, such as colour, texture and shape. These are necessary for the system to known, not just where an object is, but also what it is. In computer vision little attention has been paid to the problem of integrating these different sources of information in the context of a "seeing" system. However, lessons can be learnt from neuroscience and psychophysics. An understanding of the human vision system, may give us ideas about how the problem could be solved. In the first part of this section some relevant work done within these fields will thus be reviewed.

Even if each cue is interesting in itself and many experiments can be performed on single cues in isolation, combinations of multiple cues are necessary for a system to continuously operate in a real environment and everyday situations. A natural question is thus how different cues ought to be combined, utilizing the strength of each individual cue, while overcoming their respective weaknesses. In order for visual objects to be uniquely recognized, features from different cues has to be used in conjunction. In cognition this problem is known as the "binding problem". Studies of single cell firing rates have shown that the different low-level areas in visual cortex consist of cells specialized for different stimuli and organized topographically. Thus the creation of feature maps, such as those in the presented system, is not just natural from a computational point of view, but also biologically plausible. Within psychophysics reaction times and the ability to discriminate between different groups of features have been measured experimentally. From these experiments it has been shown that objects are much easier attended to if they are characterized by a unique feature, rather then a unique combination of features.

In a very influential paper, Treisman & Gelade (1980) presented a theory of human visual attention, Feature Integration Theory, in which this behaviour could be explained. The theory is based on the notion of low-level feature maps and an attentional spotlight. An observer is not aware of a certain feature until the feature is covered by the spotlight. The direction of the spotlight is guided by the feature maps and if a unique target feature is requested, the spotlight moves directly towards the location in question, if there is only one such feature in view. This procedure is called parallel search, since the time required does not depend on the size of the feature maps. On the other hand, conjunctions of different target features are assumed not to be processed preattentively, which means that they can only be found once the spotlight has arrived to the position of such a conjunction. Thus the feature maps have to be searched serially, with the spotlight moving randomly across the field of view, which means that the latency increases for larger feature maps.

The Feature Integration Theory soon turned out to have a number of weaknesses and had to be revised (Treisman & Sato 1990). Additional psychological experiments had shown that the distinction between pure parallel and serial search was not as clear as the theory seemed to suggest. For example, it was shown that conjunctions of disparity and colour could be found in parallel. In order to overcome these problems, Wolfe et al. (1989) proposed a model called Guided Search, in which a conjunction search is performed directly using the feature maps, similar to parallel search (Cave & Wolfe 1990). However, since the maps are supposed to be noisy, conjunctions might be less distinct and only found after a few trials. For example, finding a red cross among red circles and green crosses, is harder than finding a red cross among green ones only. The preattentive cues guide the spotlight from one region of interest to another, in the order of strongest activation and due to noise a distractor may well be stronger than the target conjunction. A similar model was presented by Duncan & Humphreys (1989), but they also used the dissimilarity between non-target features to explain variances in search performance. If a background of non-target features is homogeneous, the non-target features may be grouped and the true target is easier to find.

8.3.1 Integrating top-down and bottom-up

Today there are well established models for bottom-up attention in humans. As mentioned previously, bottom-up cues are necessary in order for the observer to react to unexpected events in the scene. It is typically assumed that a series of feature maps are created, each feature map representing a certain aspect of the visual input. These maps are then combined into a saliency map, from which an image position is found using a process of winner-takes-all (Koch & Ullman 1985). A question is whether these models can by used for an artifical system. Computationally, the saliency map is typically created as a weighted sum of all available feature maps, which means that it represents the collective activity of all maps. The feature maps are often normalized so as to promote saliency maps with a few distinct peaks (Itti et al. 1998).

Top-down information, that is information based on the will and the memory of the observer, should also be incorporated into the system. It is not as straight-forward as one might assume, since bottom-up saliency is based on how much a certain feature stands out in relation to neighbouring features, while top-down saliency is typically based on the existence of the feature itself. One way of introducing top-down control is by changing the feature weights, so that requested features have larger weights than features less relevant to the task at hand. This approach might be dangerous, since it easily makes the system either purely top-down or bottom-up. Milanese et al. (1994) instead used a non-linear relaxation process with a series of energy functions, together with a separate alerting system, that is able to bypass the results from the saliency map.

It is also possible to create the saliency maps directly using a top-down model of a desired object. One such approach is a model of Rao et al. (1997), who used correlations of a memorized model with incoming data. One advantage of such an approach is the fact that the model might require not just the existence of a feature, but also its expected strength. However, bottom-up information is only used in learning the model and not for visual search. This means that an observer cannot concentrate on one task and still react to other unexpected events. Some researchers have even suggested models with two mechanisms running in parallel, one fast bottom-up and another slower one based on top-down knowledge (Braun & Sagi 1990).

In the system presented here the summation of feature maps is done over outputs from a series of adjustable functions, with one function for each feature map. The filters can be seen as log-likelihood functions of the image position in question being an object of interest. This will be made clearer in the following section. Using ordinary summation these saliency functions would have been monotonically increasing, which means that larger feature values will always be more salient. Since the function may be changed arbitrarily depending on the task at hand, it is possible for multiple feature values to generate the same amount of saliency. In the case of binocular disparities, an observer might be able to concentrate on regions at a certain depth and at the same time be able to react to objects popping-up just in front of the view. More explicitly, the saliency at position (x, y) may be written as

$$S_{x,y} = \sum_i s_i(v^i_{x,y}), \tag{8.3}$$

where $v^i_{x,y}$ is the feature value of the ith feature map and $s_i(v)$ the corresponding saliency function. The dimensionality of feature values might be different for each feature map. For example, colour could be represented in two dimensions, using one blue-yellow channel and one green-red channel (Nordlund & Eklundh 1999).

Example: Attentional pursuit

One of the benefits of the saliency functions mentioned above, is that they may be learned and stored either in a short-term or long-term memory as a representation of an observed object. In this section it will be shown how saliency functions may be used to follow an object moving in the scene using a short-term memory of disparity and optical flow distributions. The results are similar to those of Section 7.4, but the complexity of the method is considerably lower.

Assume that a feature value v is given at a particular image point. Using Bayes' rule it is possible to express the conditional probability, for the point being part of an object of interest, as

$$P(f|v) = \frac{P(v|f)P(f)}{P(v)} = \frac{P(v,f)}{P(v)}. \tag{8.4}$$

From this function the saliency function will be given by $s(v) = \log P(f|v)$ and if there are multiple cues, these may be integrated as shown in Equation 8.3. It is easily seen that, if the distribution of feature values is assumed to be the same between two instances in time, $P(v,f)$ and $P(v)$ can be approximated using two histograms, one for all image points and another only for points within the object of interest.

The images of Figure 8.2 show a short sequence of images based on this principle. The lightly shaded areas denote image pixels for with the probability of representing the object of interest is higher that 50%. Saliency functions are created for disparities as well as optical flow values and updated from one frame to the next. The histograms of the first frame is based on a rectangular area in the upper part of the walking person. This area was found using the independent motion detection approach presented in Section 7.4. Since probabilities exist for each pixel, the histograms can be created using sums of these probabilities, rather than integer values. The results may be further improved, using the maximum a posteriori estimate of the segmentation, as will be described in the next section. Another possibility that has shown to be advantageous, is representing the object as a binary mask and updating the mask as time passes (Maki et al. 2000).

8.4 Figure-ground segmentation

In many situations one would like to determine if an image region satisfies one of two possible hypotheses. This is typically the case in figure-ground segmentation, where each pixel either belongs to the foreground or background. Another example of two such hypotheses is whether a region is part of an independently moving object or not. Let $f_i \in \{0,1\}$ be a parameter that determines which state an image pixel $\mathbf{p}_i$ belong to. The segmentation of the whole image will then be described by $\mathbf{f} = (f_1, f_2, ..., f_N)$, where N is the number of pixels. Further assume that a measurement g_i is available for each pixel. In order to estimate the true segmentation one might find the maximum a posteriori (MAP) estimate, utilizing possible dependencies between neighbouring pixels. Given the

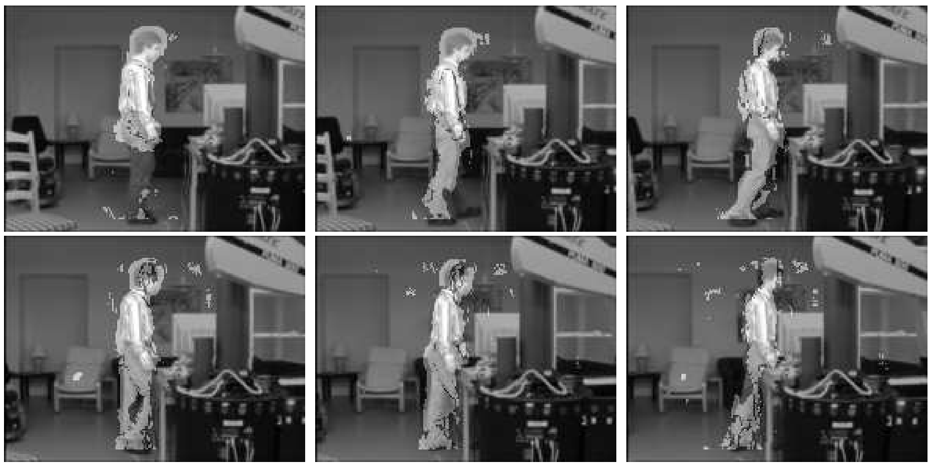

Figure 8.2. Pixels for which the probability of an object of interest is greater than 50%, based on prior distributions of disparity and optical flow.

measurement, the posterior distribution for a segmentation $\mathbf{x}$ is given by Bayes' rule,

$$P(\mathbf{x}|\mathbf{g}) = k\, P(\mathbf{g}|\mathbf{x})P(\mathbf{x}), \tag{8.5}$$

where k is a normalization constant.

The prior distribution can be hard to determine, but a possible model may be based on the similarities between neighbouring pixels. For example, in figure-ground segmentation discontinuities in depth are typically accompanied by large image gradients. Thus image gradients may be used to describe differences in discontinuity probability. If $c_{i,j}$ is a constant based on such gradients and N_i denotes the neighbourhood of $\mathbf{p}_i$, one can use the prior distribution

$$P(\mathbf{x}) = k_1 \exp\left(\sum_{i=1}^{N}\sum_{j\in N_i} c_{i,j}\,\delta(x_i, x_j)\right). \tag{8.6}$$

The function $\delta(x_i, x_j)$ is equal to 1 if $x_i = x_j$, otherwise 0. It should be pointed out that other cues may be used in conjunction with image gradients, for example disparities and colour. Given the measurement $\mathbf{g}$, the likelihood function of $\mathbf{x}$ may be expressed as

$$P(\mathbf{g}|\mathbf{x}) = \prod_{i=1}^{N} P(g_i|x_i) = \prod_{i=1}^{N} P(g_i|1)^{x_i} P(g_i|0)^{1-x_i}, \tag{8.7}$$

which can be rewritten as

$$P(\mathbf{g}|\mathbf{x}) = k_2 \exp\left(\sum_{i=1}^{N}\lambda_i x_i\right), \quad \text{where } \lambda_i = \log(\frac{P(g_i|1)}{P(g_i|0)}) \tag{8.8}$$

and $k_2 = \prod_{i=1}^{N} P(g_i|0)$ is a constant given by the measurements alone. The posterior distribution will thus be given by

$$P(\mathbf{x}|\mathbf{g}) = k_3 \exp\left(\sum_{i=1}^{N} \lambda_i x_i + \sum_{i=1}^{N} \sum_{j \in N_i} c_{i,j}\, \delta(x_i, x_j)\right). \tag{8.9}$$

8.4.1 Graph cuts

The problem of maximizing Equation 8.9, that is determining the MAP estimate $\mathbf{x}_{MAP} = \arg\max P(\mathbf{x}|\mathbf{g})$, can be solved using graph cuts. Graph methods (Ahuja et al. 1993) are well understood and lots of effort has been spent on trying to optimize the complexity of such methods. It will be shown in Section 8.4.2 that figure-ground segmentation based on graph cuts can in fact be performed in real-time. Let $G = (V, E)$ be a graph defined by a set of nodes V and a number of connections E between pairs of nodes. The nodes consist of all image pixels, together with a source node s and a drain t. All pixel nodes are connected to their respective neighbours, as well as to either s or t depending on the sign of λ_i, which is the factor defined in Equation 8.8. This means that source and drain can be seen as representing the two hypothesis, foreground or background.

Each connection has a dedicated non-negative capacity. For connections between two neighbouring pixels the capacity is $2c_{i,j}$, which may be determined using image gradients, as mentioned earlier. The factor of 2 will be explained later on. For image nodes connected to s, which is the case if $\lambda > 0$, the capacity is determined by $c_{s,i} = \lambda$. The remaining image nodes will then be connected to the drain t, with a corresponding capacity of $c_{i,t} = -\lambda_i$. Assume that the nodes have been partitioned into one set V_s, that includes s, and its compliment $V_t = V \setminus V_s$, which contains t. Such a partition is known as a graph cut.

A cut may be considered as a segmentation, with pixels within V_s having the state $x_i = 1$, while $x_i = 0$ for the remaining pixels. The total capacity for all edges that connect nodes on either side of the cut will then be given by

$$\begin{aligned} c(\mathbf{x}) &= \sum_{x_i=0} c_{s,i} + \sum_{x_i=1} c_{t,i} + \sum_{x_i=1, x_j=0} 2c_{i,j} \qquad (8.10) \\ &= \sum_{\lambda_i>0} (1 - x_i)\lambda_i + \sum_{\lambda_i<0} x_i(-\lambda_i) + \sum_{i=1}^{N} \sum_{j \in N_i} c_{i,j}(1 - \delta(x_i, x_j)) \\ &= k_4 - (\sum_{i=1}^{N} x_i \lambda_i + \sum_{i=1}^{N} \sum_{j \in N_i} c_{i,j} \delta(x_i, x_j)). \end{aligned}$$

The factor of 2 in the last term of Equation 8.10 disappears, since the following two rows include summations performed in both directions. From a comparison between the last row and Equation 8.9, it may be concluded that $\mathbf{x}_{MAP}$ can be found as the cut that minimizes $c(\mathbf{x})$. The two problems are equivalent.

Finding a minimum cut through the network, such that s and t are on either side of the cut, is possible using a maximum flow approach. This can be understood as follows. Assume that flow of some substance is led from the source to the drain, where the maximum flow along each edge is limited by the corresponding capacity. A direct path from s to t is called an augmenting path. One of the earliest methods for finding the maximum flow over a whole network, is based on a gradual search for such paths and is thus called the Augmented Path algorithm (Ford & Fulkerson 1956). After an augmenting path has been found, it is filled up with the maximum possible flow, which is limited by the smallest capacity along the path. In the end the network will be saturated and no additional flow can be led through, since all possible paths will be blocked by at least one edge for with the flow has reached the capacity. A cut through the network will now be given by the capacities of these saturated edges. Since the flow will always be limited by the cut of lowest collected capacity, this cut is in fact the minimum cut, as defined above.

The most popular method currently in use is known as the preflow-push algorithm by Goldberg & Tarjan (1986) and is based on flow being pushed from the source towards the drain. Flow that has not yet reached the drain is called excess flow. Each node includes a record that determines the shortest distance to the drain. Starting from the node with the greatest distance to the drain, excess flow is gradually pushed closer to the drain, using directions defined by the distance labels of neighbouring nodes. This means that flow from different directions is gathered up before being sent to the drain, which typically results in a much faster execution. Since the amount of flow that may be pushed across an edge is limited by the edge capacity, some excess flow might remain unpushed. For these nodes the distance labels are updated using the labels of neighbouring nodes. Since flow is always pushed from the longest distance, this flow will eventually be tested in a new direction. The process stops when no additional excess flow can be pushed any further.

An illustration of a network before and after flow has been traversed along an augmenting path can be seen in the upper and lower images of Figure 8.3. The circles show image pixels, with capacities based on their different colours. Even if the connections between image nodes are shown as undirected, they are implemented using two directed edges. Thus flow in one direction may be canceled out by an equal amount of flow in the opposite direction. The upper arrows indicate the capacity of connections from the source, while the lower ones point towards the drain. In the lowest image the residual capacities have been reduced by the amount defined by the lowest capacity along the augmenting path. In the current example no additional paths can be found. A saturated edge has been introduced, which separates the nodes depending on their connections to source and drain.

The complexity of the preflow-push algorithm is $O(n^2 \log n)$, where n is the number of pixels. However, due to the regular structure of typical computer vision applications, such a complexity is rarely seen in practice. In the case of integer capacities the best maximum flow method to date is an approach of

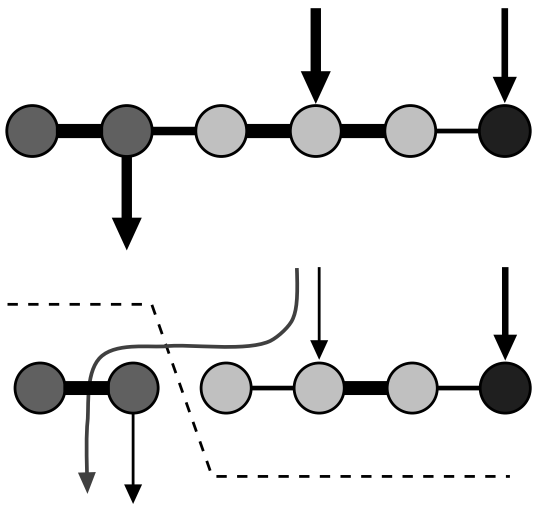

Figure 8.3. A network with residual capacities before (upper) and after (lower) flow has been led from source to drain through an augmenting path.

Goldberg & Rao (1997). Instead of using distance labels representing the number of edges that has to be traversed in order to reach the drain, the method relies on an adaptive binary length function. This length function is based on the current residual capacities and a threshold that is gradually decreased such that edges of large residual capacities will be treated first. The complexity of such an approach is $O(n^{3/2} \log n)$.

Within computer vision graph cuts have been used for applications such as image restoration and disparity calculation. Greig et al. (1989) compared the results of a maximum flow based method to simulated annealing for restoration of binary images. Even if the results were improved, they also concluded that a MAP estimate is not necessarily the best method. The MAP estimate might in fact be a poor representation of the whole posterior distribution (Fox & Nicholls 2000). For disparities Roy & Cox (1998) used a network with d nodes stacked on-top of each other for every image pixel. The two sides of such a columns are connected to the source and drain respectively. The calculated disparity of a certain pixel is determined by the location of the cut along the corresponding column. As a result the cost of a discontinuity will be linearly dependent on the difference in disparity. A similar network has been used for image restoration on grey-scale images (Ishikawa & Geiger 1999).

Unfortunately, it is sometimes hard to represent the prior distribution using such a cost function. A better cost function could be one for which the discontinuity costs are kept constant, instead of letting it grow for larger differences in disparity. A method of Boykov et al. (1998) is based on such a cost function. Instead of just two terminals, source and drain, their network contains one terminal for each disparity and disparities are determined using a multi-way cut approach. The image nodes are clustered such that each node is connected to one terminal, with cuts between every cluster. The problem is solved iteratively with a maximum flow found between every pair of terminals. This method is slow, but the results are impressive.

In this study we have used graph cuts for figure-ground segmentation, based on either image differencing or zero disparities. In order to use the full potential of the method, also object specific information should be used. Since this study is limited to stereo and motion cues, this graph cut based method has not yet been integrated into the rest of the system. However, an example will be shown in the next section, illustrating the expected performance.

8.4.2 Example: Image differencing

In Chapter 3 sequences of images were stabilized with respect to rotation and in the end consecutive images were subtracted, generating residual images from which moving scene objects could be found. An example of such a procedure can be seen in the second row of Figure 8.4. The images show regions where the residual exceeds a threshold of 8 in image pixel values. In order to pinpoint the exact location of the object in the image, a blob detector is typically employed on the residual. However, in a case like this, where data are only available around image gradients, the data are quite sparse, which leads to a number of small blobs being found, instead of just one for each moving object. This is a perfect example where one would benefit from a filling-in method based on graph cuts.

For the experiment presented in this section a very simple model was used. A pixel is assumed to be part of a moving object in the foreground, if the residual $|I_t(i)|$ is larger than 8, assuming that the maximum luminance value is 255. The network nodes representing these pixels each have one connection to the source, but no connections to the drain. The opposite is true for pixels of small residuals. The larger the residual, the more certain it is that the pixel really belongs to the foreground. This is reflected by the edge capacities, which are defined by

$$c_{s,i} = 20\,|I_t(i)|. \tag{8.11}$$

A small residual does not necessarily mean that the pixel is a part of the stabilized background, since it could also be due to the foreground being uniformly shaded. However, for regions of large image gradients, such a confusion is less likely. This means that the background edge capacities should preferably take the gradients into consideration. This is done using the following capacities

$$c_{i,t} = |I_{xx}(i) + I_{yy}(i)|. \tag{8.12}$$

All capacities and flow values are represented used short integers, in order to keep the memory consumption low. The constants used for the definitions of capacities were found empirically for this particular problem, but are irrelevant for a general understanding of the method. The capacity of edges between two

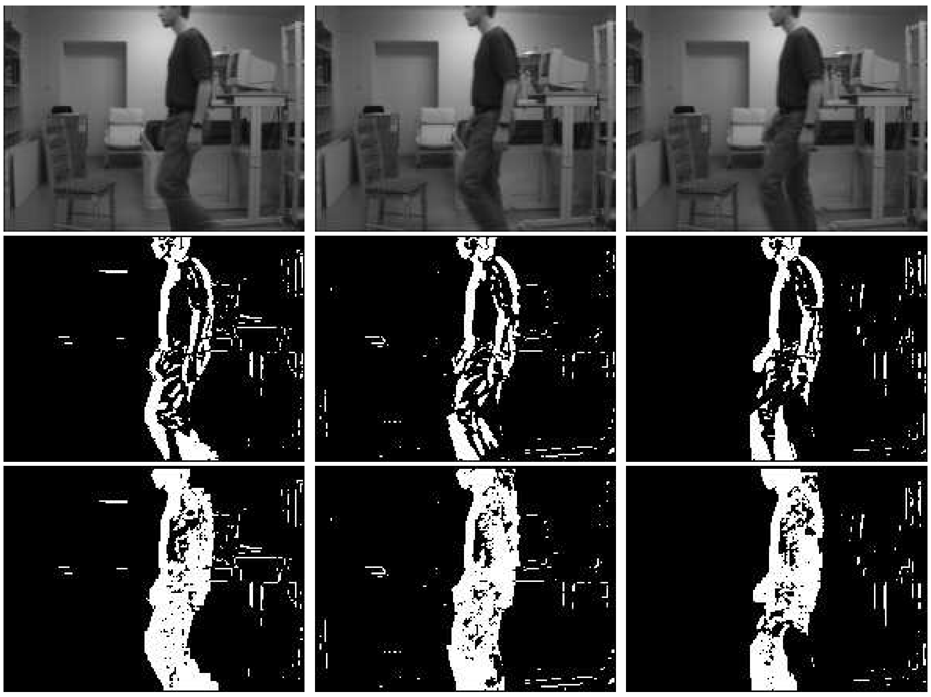
Figure 8.4. Subtraction and thresholding (middle) of consecutive images from a sequence of incoming images (upper), with the results after performing a filling-in operation based on graph cuts (lower).

pixel nodes determines the amount of cooperation that is possible between the nodes. In this example the capacities are defined as a function of image gradients,

$$c_{i,j} = 8\ \max(32 - |I(i) - I(j)|, 0). \tag{8.13}$$

Near gradients larger than 32, the segmentation is determined directly by $c_{i,t}$ and $c_{s,i}$, while the remaining pixel states are based on information from nearby image regions. Flow will spread from location of high confidence to uniformly shaded areas in-between. This can be seen in the lower row of Figure 8.4. From these images it is much easier to find the exact image location of the moving object than before textureless areas were filled in.

The computational cost, which is approximately 25 ms for a 192×144 pixel image, depends on the number of pixels being filled and the distance between source and drain, that is the number of pixels between a foreground pixel in the lower row of the figure and the closest foreground pixel before filling-in. Thus the speed is much higher than what the algorithm complexity would suggest. The same method has been used for a number of different problems, such as improving the output of a zero disparity filter. However, it is not clear where in the flow of information this operation ought to be performed. It could either be done prior to cue integration on each cue individually or immediately after using the combined saliency map.

8.5 Conclusions

This chapter presented the integration of the cues and operations described in earlier chapters into a complete system. More details were given about operations that had earlier been mentioned. The system consists of three separate feature paths; an optical flow path, a disparity path and a path for independent motion detection. It was shown how cues from these paths can be integrated in terms of saliency maps, from which information about possible regions of interest can be extracted. This presentation also included a description of how top-down knowledge can be added to the system.

The developed system has by no means all the characteristics outlined in the introduction in Chapter 1. Stereo and motion cues are strong in that they contain information about the locations and motions of objects present in the scene. However, in order for a familiar object to be recognized, additional cues such as colour, texture and local shape, also need to be included. On the other hand, a framework is created from which we claim that such information can be extracted. In order to determine the properties of a particular object, one first has to find where it is located and then what it looks like. Stereo and motion cues may provide such information, as has been shown in this chapter.

Chapter 9

Conclusions

The goal of this thesis is to develop the lowest reactive levels of an artificial system that "sees". Such as a system could e.g. be an autonomous robot using vision for tasks such as navigation, and fetching and carrying objects, that is including both detection and identification of things in the environment and physical interaction with them. In order to be successful the system would have to extract visual information sufficient for the task at hand. However, since the world is constantly changing unpredictable events may occur and the observer must adapt its behaviours accordingly. Hence, the system first has to notice that something unexpected has happened and then gather enough relevant information for it to react accordingly. Since the nature of events and objects may be unknown in advance, a whole range of different cues and methods should be considered.

The work has been concentrated on cues based on motion and binocular stereo data. These cues are especially valuable, since they give an indication of the three-dimensional extent of visible objects and how the observer is located and moving in relation to these objects. For the observer to react appropriately, such knowledge is essential. Much of the work has been biologically inspired, but not necessarily biologically plausible. The intention has not been to create a model of the human vision system, but an artificial vision system that may be used by for example an autonomous robot. Every component has been evaluated in terms of its real-time use and not only on accuracy. The real-time constraint is necessary for the system to be tested in practice, when the actions of the seeing system directly affect the visual stimuli.

9.1 Summary and contributions

The subjects covered in the thesis may be divide into three different themes; that of analyzing and selecting stereo and motion algorithms for real-time use, that of developing methods for 3D visual analysis and that of integrating multiple types of visual information into a complete system. The studies presented in Chapters 2, 5 and 6 are analytical and cover a number of methods that have earlier been presented by others. Most innovations can be found in Chapters 4, 7 and 8. The most important results from each individual chapter is given as follows:

In Chapter 2, three different optical flow algorithms were evaluated and analyzed. All three methods were based on image gradients and the optical flow constraint. Two of these also included a smoothness constraint applied globally and a fast Preconditioned Conjugate Gradient method was used to solve the optical flow. Accuracy turned out to be good enough only with significant smoothing. The last algorithm included a robust version of the smoothness constraint, which led to a better preservation of discontinuities. The only method fast enough for real-time use, was the method of Lucas & Kanade, that does not have a smoothness constraint, but aggregates data only within local windows.

Optical flow can only be accurately calculated, if the magnitudes stay within the range allowed by the involved algorithms. In order to make sure that this is the case, the presented system uses image stabilization. A number of stabilization algorithms were explored in Chapter 3, one based on corner features, another using contour points and a third one working on all image points. The first method being robustly implemented, is also the most accurate, but it might fail if there are not enough corner features. The second method is a new approach and combines the strengths of the other methods. At a somewhat higher computational cost, its accuracy is close to that of the corner method, but this method is less likely to collapse.

Chapter 4 contains a large portion of the innovations in this thesis. It was shown how the epipolar geometry of a binocular stereo head can be continuously and robustly estimated in real-time. The system was constrained, so as to minimize the number of free parameters, but without limiting its flexibility and use. Unlike what could be expected, the essential matrix is not necessarily preferable to the bilinear optical flow constraint. Since the latter is just an approximation, it is usually disregarded for stereo. However, the results from this chapter show that the optical flow constraint is not only faster than methods based the essential matrix, but also more robust.

The epipolar geometry is needed by the disparity algorithms evaluated in Chapter 5. These methods rely on the epipolar lines being parallel to the scan lines, which is possible through a rectification using the epipolar geometry. Furthermore, from the corner matching in Chapter 4, the range in which disparities can be found is also given. The evaluation of disparity algorithms shows that the major difference between methods is primarily in the density of the calculated results and not in their accuracy. Since the difference in computational cost is so large, it is hard to justify the use of more complex algorithms unless the density is critical, which might be the case in reconstruction.

In Chapter 6, six different monocular structure-from-motion methods were tested, three linear and three iterative ones. The hope was that the results could be used to determine the ego-motion of the observer, as well as three-dimensional structure. However, the results were only judged accurate enough if translations were large in relation to what is typical in our application. The strengths and weaknesses of the algorithms turned out to be quite similar, which indicated that the difficulties are inherent to the problem itself.

More successful results were possible if ego-motion was instead determined binocularly, which was done in Chapter 7. A number of methods were presented with which this could be done. The best results were achieved if 3D points triangulated from stereo were matched in time, rather than when one combined stereo and motion in the same optimization. Later on in the same chapter, it was shown how these results may be used in conjunction with disparities to efficiently determine regions of independent motion.

The components presented in the previous chapters, were integrated into a complete system in Chapter 8, and the flow of information between different components was described. Most of the implementation details were explained, together with the additional components that had to be added for the system to work in practice. It was also discussed how cues could be integrated and top-down knowledge used to control the behaviour of the system.

9.2 Possible extensions

Even if the presented system is based on an extensive analysis of possible techniques, that have been implemented and integrated, it is by no means a final and fully functional system. Some possible extensions that could be considered in a future system can be summarized as follows.

Fixational mechanism The binocular system assumes that the cameras are always kept in fixation, that is they are directed towards the same point in 3D space. This mechanism has not yet been implemented, even if information required to perform such an operation is available from the extracted motion and stereo data. Techniques for performing dynamic fixation were extensively studied ten years ago using less explicit information. We are in a position to go beyond that work with the presented system.

Adaptation Most components of existing and future systems can be adapted in order to optimize performance, either short-term of long-term locally or in relation to other components of the system. They typically contain thresholds and parameters that can potentially be learned. It would be worth investigating how such mechanisms could be provided.

State based methods for gaze control Depending on the task at hand there are alternative ways of controlling the gaze. Some of these changes in behaviour can be done continuously, updating parameters involved in the process. Other changes are discrete in nature. For example, the system should either stabilize the background or foreground. This would require a state machine to be implemented.

Hierarchy of methods The same methods should not necessarily be run all the time. When little is known about the environment in which the system is working, methods usually have to be conservative and use every possible source of information. However, if the system learns more about the environment, a conservative and slow method may be changed for a faster one, by exploiting the information learnt. Different methods for similar purposes could be arranged in a hierarchy of methods, ordered based on what prior information they require. Toyama & Hager (1999) used such an approach for tracking.

Additional visual cues Motion and stereo cues contain no explicit information about particular objects and can only be used for object recognition, if information such as shape, size and rigidity is derived from them. Other cues should be considered as well, for instance cues based on colour, texture and form. Only with enough such information can a visual object be uniquely represented.

Object representation A difficult general problem is how objects are to be represented, stored in memory and retrieved when needed. It is also important to know when and how an image region is to be considered as such an object. The representation should be as general as possible, so that all kinds of objects may be treated. However, the problem of matching stored representations with incoming visual data is greatly complicated if the models are too general. These issues are far-reaching and encompass most problems on what "seeing" actually means.

Embodiment The visual system is only a part of a larger system, that of the observer itself. The observer occupies a part of the dynamic environment, with means of performing physical interaction. The visual system cannot interact on its own, only through the motor part of the observer. This requires communication in terms of information passed to and from the motor system. In order to really evaluate the performance of the visual system, it thus has to be implemented within a larger context.

We have shown that the presented system may be used as an essential component of an artificial system that "sees", in order to determine where objects are located and events occur in the environment in which the system works. Even if there is more to be done to make intelligent "seeing" systems a reality, we believe that such systems may well exist in the future.

Appendix

Appendix A

Speeding up the code

The problem of reaching real-time speed can be approached in a number of possible ways. In his book *Vision* Marr (Marr 1982) distinguished between three different levels of information processing; the computational, the algorithmical and the implementational. All of these ought to be considered. On the computational level the needs of the system are defined. What has to be performed? And what does not? For example, if it is not absolutely necessary for a system to calculate and store the gradients of incoming images, one should avoid it. Typically gradients are needed, but they not always have to be stored. Designers tend to concentrate on optimizing the actual calculations, but underestimate the cost of storing. Towards the end of this appendix we will show that memory optimization is just as important.

From a designer's point of view, it is often easier to jump directly to the second level, the algorithmical one. Modeling the desired behaviour of the whole system can sometimes be too overwhelming and forces you to concentrate on it piece by piece. But once the puzzle has been assembled, one might benefit from going back to the computational level and reconsider some aspects of the design. If you want your system to be able to perceive image motion, it is natural to look for optical flow algorithms, but once optical flow is available, what is it really used for? Could the same problem be solved without a dense flow field, and does it matter if it is forgotten once it has been calculated and used? On the way between incoming data and final actions, what is necessary and what can be dropped? Cutting all loose ends is an important part of the gradual development of an information system.

The algorithmical level deals with the processes to be used for solving a particular problem and how data should be represented, while the implementational one describes how and on what platform the algorithms are to be realized. It is on these levels that most design efforts are typically spent. Quite often the hardware on which the implementation is to be made is given and the search for algorithms can be restricted to those for which the hardware is adequate. A general purpose computer might then be a better alternative than dedicated hardware, since such hardware may seriously constrain the search for possible algorithms and representations. Hardware should be evaluated based on algorithms already known, as well as on methods that has yet to be invented.

Assume that a working system already exists, but is still not running in real-time, that is fast enough in relation to changes in the environment in which it is supposed to work. Then there are a number of obvious likely solutions to consider, such as those given in the list below. Before you listen to the cries for faster hardware, the search for other alternatives ought to be exhausted. Even if faster hardware is an obvious choice, it is well worth knowing why the existing hardware failed to reach its expectations. This is no easy task, since the lack of pure computational power is not always the explanation. The problem might be in the usage of memory, in the interaction with external devices or due to the fact that too much power is spent on operations that turn out not to be necessary. In essence, in order to reach this understanding, the remaining ways of improvements are preferably evaluated first.

Means of gaining speed

- Faster hardware
- Lower resolution
- Lower frame-rate
- Faster algorithms
- Faster implementation
- ...

A lowered resolution is something that often show to be more promising, than what one might expect. The success of different algorithms is often judged based on how we perceive the output, often the in terms of images showing information such as optical flow or binocular disparities. However, such images are rarely what constitutes an end result and what really counts is in what sense the system is successful in its tasks. For example, if optical flow is to be used for tracking, a lowered resolution by half in each dimension and a double frame-rate, might show more success than would otherwise have been the case, even if the computational cost is reduced. The same argument can be applied when evaluating different algorithms, in that there might be a trade-off between frame-rate and the complexity of algorithms. In order to maximize the speed through a faster implementation, the strengths and weaknesses of the available hardware need to be understood, as the next section will show.

A.1 The cost of memory

That the usage of memory is indeed costly can be learned, if one analyzes the latency of single read and write operations as a function of the total amount of memory in use. All modern computers of today, such as the 1.2 GHz Athlon

MP processor exemplified here, uses high-speed cache memories to improve the memory latency of frequently accessed data. Since larger memories are typically slower and smaller memories can be located closer to the processor, cache memories tend be limited in size. This means that the total amount of data in use should be kept within the sizes of the caches, in order to maximize speed. For example, an Athlon MP processor has normally got a memory structure with two levels of cache memories. An access beyond the 128Kb L1 cache costs around 18 clock-cycles, for both read and write operations. If a memory access misses the L1 cache as well as the 256Kb L2 cache, the cost increases dramatically. A write operation requires about 100 extra cycles, whereas a read costs about half. This means that it is often wiser to recompute data than storing it, even if the results are used multiple times in different parts of the process.

Since cycles are lost due to cache misses only once every new 64 byte cache block is accessed, it is better to take advantage of the fact that after the first bytes of a block has been accessed, reading the remaining bytes is virtually free of charge. If other memory blocks are accessed in between, there is a risk that the block will otherwise be swapped out. So, if it is known that cache misses cannot be prevented, the data are preferably accessed block-wise and not in an order such that many different cache blocks are accessed simultaneously. Typically when an image processing operation is performed on entire images, this is not much of a problem, since it is natural to access data sequentially. However, if data are store in data structures with a number of different variables for each instance, it is quite often the case that only a few variables are being accessed, while the others remain untouched. It might then be preferable to break the structures into different substructures, each consisting of variables that are accessed simultaneously.

A.2 A blurring example

Suppose we are interested in blurring an incoming image, using an ordinary 5×5 Gaussian filter kernel. What you typically do is first lowpass filter the image x-wise, store the result in a temporary image structure, and then perform the same operation y-wise. If the final result is stored in another structure, you end up using three different images. If the images are 320×256 in size and stored using floating points, we have already used close to 1Mb of memory, which is more than the capacity of the caches. Assume instead that the incoming and resulting images are stored as characters and the temporary image as short integers, the amount of memory has been reduced to 320Kb. If the incoming image is never to be reused, the same memory area can be used for incoming as well as resulting image. The only annoying thing left is the temporary buffer.

In the work presented here, the need for such temporary images is minimized using rotating buffers, instead of whole images. Each operation, such as lowpass filtering x-wise and y-wise, is performed pixel row by pixel row. As soon as 5 pixels rows have been filtered x-wise and stored in a temporary buffer, it is possible to lowpass filter the first row of the output y-wise. The first pixel row of the buffer can then be reused for the next incoming x-wise filtered row. The example code may look as follows:

```
Image   img(width, height);
Buffer  buf(width, 5);

for (int y=0; y<4; y++) {
  LowPassXRow(img[y], buf[y]);
}
for (int y=4; y<height; y++) {
  LowPassXRow(img[y], buf[y]);
  LowPassYRow(buf[y], buf[y-1], buf[y-2],
    buf[y-3], buf[y-4], img[y-2]);
}
```

where `operator[ ]` is overloaded, such that the `y mod 5` row of `buf` is returned. Each row routine is very simple in complexity, which makes them easy to optimize. For AMD and Intel processors an additional speed-up can be gained, implementing the routines using SIMD (Single Instruction Multiple Data) instructions, that are available for these processors. Most row routines used in this study have been implemented using MMX instructions, that are able to perform 4 short integer or 8 character operations in parallel. For the given example the following table summarizes the computational cost of blurring an incoming image, based on the representation of temporary data, with all pixel values stored as short integers.

Buffer type	Cost (ms)
Entire image	10.1
Rotating (C)	5.9
Rotating (MMX)	0.8

Figure A.1. Cost of blurring a 320 × 256 image

It is worth noting that the speed-up thanks to the MMX implementation is in fact greater than what the 4-way parallelism might imply. There is a rather easy explanation of this. When MMX operations are used, calculations will be done using the MMX registers, while the integer registers will be dedicated for loop counters and memory pointers, but without MMX the same integer registers have to be used for all kinds of data. An implementation of the Harris corner detector using the same principle runs in about 3.0 ms and with a search range of 16 pixels, binocular disparities are calculated in 11.1 ms.

Bibliography

Adelson, E. & Bergen, J. (1985), 'Spatiotemporal energy models for the perception of motion', *Journal of the Optical Society of America* A **2**(2), 284–299.

Adiv, G. (1985*a*), 'Determining 3-d motion and structure from optical flow generated by several moving objects', *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(4), 384–401.

Adiv, G. (1985*b*), Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Francisco, CA, pp. 70–77.

Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993), *Network flows: Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, NJ.

Anandan, P. (1989), 'A computational framework and an algorithm for the measurement of visual motion', *Intl. Journal of Computer Vision* **2**(3), 283–310.

Andersson, M., Orebäck, A., Lindström, M. & Christensen, H. (1999), *Sensor Based Intelligent Robots*, Springer, Berlin, chapter ISR: An Intelligent Service Robot, pp. 287–310.

Arun, K., Huang, T. & Blostein, S. (1987), 'Least-squares fitting of two 3-d point sets', *IEEE Trans. Pattern Analysis and Machine Intelligence* **9**(5), 698–700.

Ayer, S. & Sawhney, H. (1995), Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Cambridge, MA, pp. 777–784.

Bab-Hadiashar, A. & Suter, D. (1998), 'Robust optic flow computation', *Intl. Journal of Computer Vision* **29**(1), 59–77.

Baker, H. & Binford, T. (1981), Depth from edge and intensity based stereo, *in* 'Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Vancouver, Canada, pp. 631–636.

Balakirsky, S. & Chellappa, R. (1996), 'Performance characterization of image stabilization algorithms', *Real-Time Imaging* **2**(5), 297–313.

Ballard, D. (1991), 'Animate vision', *Artificial Intelligence* **48**(1), 57–86.

Barnard, S. (1989), 'Stochastic stereo matching over scale', *Intl. Journal of Computer Vision* **3**(1), 17–32.

Barron, J., Fleet, D. & Beauchemin, S. (1994), 'Performance of optical flow techniques', *Intl. Journal of Computer Vision* **12**(1), 43–77.

Beauchemin, S. & Barron, J. (1995), 'The computation of optical-flow', *ACM Computing Surveys* **27**(3), 433–467.

Belhumeur, P., Kriegman, D. & Yuille, A. (1997), The bas-relief ambiguity, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Juan, PR, pp. 1060–1066.

Belhumeur, P. & Mumford, D. (1992), A bayesian treatment of the stereo correspondence problem using half-occluded regions, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Champaign, IL, pp. 506–512.

Bergen, J. & Adelson, E. (1987), 'Hierarchical, computationally efficient motion estimation algorithm', *Journal of the Optical Society of America* A **4**(35).

Bergen, J., Anandan, P., Hanna, K. & Hingorani, R. (1992), Hierarchical model-based motion estimation, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Santa Margherita Ligure, Italy, pp. 237–252.

Bergen, J., Burt, P., Hingorani, R. & Peleg, S. (1992), 'A three-frame algorithm for estimating two-component image motion', *IEEE Trans. Pattern Analysis and Machine Intelligence* **14**(9), 886–896.

Birchfield, S. & Tomasi, C. (1998), 'A pixel dissimilarity measure that is insensitive to image sampling', *IEEE Trans. Pattern Analysis and Machine Intelligence* **20**(4), 401–406.

Black, M. (1992), Robust Incremental Optical Flow, PhD thesis, Department of Computer Science, Yale University.

Black, M. (1994), Recursive non-linear estimation of discontinuous flow fields, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Stockholm, Sweden, pp. A:138–145.

Black, M. & Anandan, P. (1993), A framework for the robust estimation of optical flow, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Berlin, Germany, pp. 231–236.

Black, M. & Jepson, A. (1994), Estimating multiple independent motions in segmented images using parametric models with local deformations, *in* 'IEEE Workshop on Non-rigid and Articulate Motion', Puerto Rico, pp. 220–227.

Black, M. & Jepson, A. (1996), 'Estimating optical-flow in segmented images using variable-order parametric models with local deformations', *IEEE Trans. Pattern Analysis and Machine Intelligence* **18**(10), 972–986.

Blake, A. & Zisserman, A. (1987), *Visual Reconstruction*, The MIT Press, Cambridge, Massachusetts.

Bober, M. & Kittler, J. (1994), Robust motion analysis, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Seattle, WA, pp. 947–952.

Bolles, R., Baker, H. & Hannah, M. (1993), The JISCT stereo evaluation, *in* 'Proc. DARPA Image Understanding Workshop', Washington, DC, pp. 263–274.

Boykov, Y., Veksler, O. & Zabih, R. (1998), Markov random fields with efficient approximations, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Santa Barbara, CA, pp. 648–655.

Boykov, Y., Veksler, O. & Zabih, R. (2001), 'Fast approximate energy minimization via graph cuts', *IEEE Trans. Pattern Analysis and Machine Intelligence* **23**(11), 1222–1239.

Brandt, J. (1997), 'Improved accuracy in gradient based optical flow estimation', *Intl. Journal of Computer Vision* **25**(1), 5–22.

Braun, J. & Sagi, D. (1990), 'Vision outside the focus of attention', *Perception and Psychophysics* **48**, 45–58.

Brooks, M., de Agapito, L., Huynh, D. & Baumela, L. (1996), Direct methods for self-calibration of a moving stereo head, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Cambridge, UK, pp. II:415–426.

Brown, L. (1992), 'A survey of image registration techniques', *ACM Computing Surveys* **24**(4), 325–376.

Bruss, A. & Horn, B. (1983), 'Passive navigation', *Computer Vision Graphics and Image Processing* **21**(1), 3–20.

Burt, P. & Julesz, B. (1980), 'A disparity gradient limit for binocular fusion', *Perception* **9**, 671–682.

Camus, T. (1997), 'Real-time quantized optical flow', *Real-Time Imaging* **3**, 71–86.

Canny, F. J. (1986), 'A computational approach to edge detection', *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**(6), 679–698.

Cave, K. & Wolfe, J. (1990), 'Modeling the role of parallel processing in visual search', *Cognitive Psychology* **22**, 225–271.

Chiuso, A., Brockett, R. & Soatto, S. (2000), 'Optimal structure from motion: Local ambiguities and global estimates', *Intl. Journal of Computer Vision* **39**(3), 195–228.

Cochran, S. & Medioni, G. (1992), '3-d surface description from binocular stereo', *IEEE Trans. Pattern Analysis and Machine Intelligence* **14**(10), 981–994.

Coombs, D. & Brown, C. (1993), 'Real-time binocular smooth-pursuit', *Intl. Journal of Computer Vision* **11**(2), 147–165.

Cox, I., Hingorani, S., Naggs, B. & Rao, S. (1992), Stereo without disparity gradient smoothing: A bayesian sensor fusion solution, *in* 'Proc. British Machine Vision Conference (BMVC)', Leeds, UK, pp. 337–346.

Cox, I. J., Hingorani, S. L., Rao, S. B. & Maggs, B. M. (1996), 'A maximum likelihood stereo algorithm', *Computer Vision and Image Understanding* **63**(3), 542–567.

Daniilidis, K. & Nagel, H. (1990), 'Analytic results on error sensitivity of motion estimation from two views', *Image and Vision Computing* **8**(4), 297–303.

Daniilidis, K. & Thomas, I. (1996), Decoupling the 3d motion space by fixation, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Cambridge, UK, pp. I:685–696.

Darrell, T. & Pentland, A. (1991), Robust estimation of a multi-layer motion representation, *in* 'Proc. Workshop on Visual Motion', Princeton, NJ, pp. 173–178.

Davis, L., Bajcsy, R., Herman, M. & Nelson, R. (1996), RSTA on the move: Detection and tracking of moving objects from an autonomous mobile platform, *in* 'Proc. ARPA Image Understanding Workshop', Palm Springs, CA, pp. 651–664.

de Agapito, L., Huynh, D. & Brooks, M. (1998), Self-calibrating a stereo head: An error analysis in the neighbourhood of degenerate configurations, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', pp. 747–753.

de Micheli, E., Torre, V. & Uras, S. (1993), 'The accuracy of the computation of optical flow and the recovery of motion parameters', *IEEE Trans. Pattern Analysis and Machine Intelligence* **15**(5), 434–447.

Deriche, R., Kornprobst, P. & Aubert, G. (1995), Optical flow estimation while preserving its discontinuities: A variational approach, *in* 'Proc. Asian Conf. on Computer Vision (ACCV)', Singapore, pp. 290–295.

Dhond, U. & Aggarwal, J. (1989), 'Structure from stereo: A review', *IEEE Trans. Systems, Man and Cybernetics* **19**(6), 1489–1510.

Dickmanns, E. (1997), Vehicles capable of dynamic vision, *in* 'Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Nagoya, Japan, pp. 1577–1592.

Duncan, J. & Humphreys, G. (1989), 'Visual search and stimulus similarity', *Psychological Review* **96**(3), 433–458.

Duric, Z. & Rosenfeld, A. (1996), 'Image sequence stabilization in real-time', *Real-Time Imaging* **2**(5), 271–284.

Faugeras, O., Hotz, B., Mathieu, H., Vieville, T., Zhang, Z., Fau, P., Theron, E., Moll, L., Berry, G., Vuillemin, J., Bertin, P. & Proy, C. (1993), Real-time correlation-based stereo: algorithm, implementation and application, Technical Report 2013, INRIA Sophia Antipolis.

Fennema, C. & Thompson, W. (1979), 'Velocity determination in scenes containing several moving objects', *Computer Graphics Image Processing* **9**, 301–315.

Fermüller, C. & Aloimonos, Y. (1993), 'The role of fixation in visual-motion analysis', *Intl. Journal of Computer Vision* **11**(2), 165–186.

Fermüller, C., Shulman, D. & Aloimonos, Y. (2001), 'The statistics of optical flow', *Computer Vision and Image Understanding* **82**(1), 1–32.

Fischler, M. A. & Bolles, R. C. (1981), 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM* **24**(6), 381–395.

Fleet, D. & Jepson, A. (1990), 'Computation of component image velocity from local phase information', *Intl. Journal of Computer Vision* **5**(1), 77–104.

Fleet, D., Jepson, A. & Jenkin, M. (1991), 'Phase-based disparity measurement', *Computer Vision Graphics and Image Processing* **53**(2), 198–210.

Ford, L. R. & Fulkerson, D. R. (1956), 'Maximal flow through a network', *Canadian Journal of Mathematics* **8**, 399–404.

Fox, C. & Nicholls, G. (2000), Exact map states and expectations from perfect sampling: Greig, Porteous and Seheult revisited, Technical report, Department of Mathematics, Auckland University.

Fua, P. (1993), 'A parallel stereo algorithm that produces dense depth maps and preserves image features', *Machine Vision Applications* **6**(1), 35–49.

Geiger, D., Ladendorf, B. & Yuille, A. (1992), Occlusions and binocular stereo, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Santa Margherita Ligure, Italy, pp. 425–433.

Gennery, D. (1980), Modelling the Environment of an Exploring Vehicle by Means of Stereo Vision, PhD thesis, Department of Computer Science, Stanford University.

Goldberg, A. V. & Rao, S. (1997), Beyond the flow decomposition barrier, *in* 'Proc. IEEE Symposium on Foundations of Computer Science', Miami Beach, FL, pp. 2–11.

Goldberg, A. V. & Tarjan, R. E. (1986), A new approach to the maximum flow problem, *in* 'Proc. ACM Symposium on Theory of Computing', pp. 136–146.

Golub, G. & Van Loan, C. (1996), *Matrix computations, 3rd edition*, The Johns Hopkins University Press, Baltimore, MD, USA.

Greig, D., Porteous, B. & Seheult, A. (1989), 'Exact maximum a posteriori estimation of binary images', *Journal of the Royal Statistical Society* **B** **51**(2), 271–217.

Grimson, W. (1981), 'A computer implementation of a theory of human stereo vision', *Proceedings of the Royal Society* **B 292**, 217–253.

Grimson, W. (1985), 'Computational experiments with a feature based stereo algorithm', *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(1), 17–34.

Hanna, K. (1991), Direct multi-resolution estimation of ego-motion and structure from motion, *in* 'Proc. Workshop on Visual Motion', Princeton, NJ, pp. 156–162.

Hannah, M. (1974), Computer Matching of Areas in Stereo Imagery, PhD thesis, Department of Computer Science, Stanford University.

Hannah, M. (1985), SRI's baseline stereo system, *in* 'Proc. DARPA Image Understanding Workshop', Miami Beach, FL, pp. 149–155.

Hansen, M., Anandan, P., Dana, K., van der Wal, G. & Burt, P. (1994), Realtime scene stabilization and mosaic construction, *in* 'Proc. ARPA Image Understanding Workshop', Monterey, CA, pp. I:457–465.

Harris, C. & Stephens, M. (1988), A combined corner and edge detector, *in* 'Proc. Alvey Vision Conf.', Manchester, UK, pp. 147–151.

Hartley, R. (1992), Estimation of relative camera positions for uncalibrated cameras, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Santa Margherita Ligure, Italy, pp. 579–587.

Heeger, D. (1987), Optical flow from spatiotemporal filters, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', London, UK, pp. 181–190.

Heeger, D. & Jepson, A. (1992), 'Subspace methods for recovering rigid motion I: Algorithms and implementation', *Intl. Journal of Computer Vision* **7**(2), 95–117.

Heel, J. (1990), Direct estimation of structure and motion from multiple frames, Technical Report AI Memo 1190, MIT AI Lab.

Horn, B. (1987*a*), 'Closed form solutions of absolute orientation using orthonormal matrices', *Journal of the Optical Society of America* A **5**(7), 1127–1135.

Horn, B. (1987*b*), 'Closed form solutions of absolute orientation using unit quaternions', *Journal of the Optical Society of America* A **4**(4), 629–642.

Horn, B. & Schunck, B. (1981), Determining optical flow, *in* 'Proc. DARPA Image Understanding Workshop', Washington, DC, pp. 144–156.

Huang, T. & Faugeras, O. (1989), 'Some properties of the *e* matrix in two-view motion estimation', *IEEE Trans. Pattern Analysis and Machine Intelligence* **11**(12), 1310–1312.

Intille, S. & Bobick, A. (1994), Disparity-space images and large occlusion stereo, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Stockholm, Sweden, pp. B:179–186.

Irani, M. (1999), Multi-frame optical flow estimation using subspace constraints, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Corfu, Greece, pp. 626–633.

Irani, M., Anandan, P. & Hsu, S. (1995), Mosaic based representations of video sequences and their applications, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Cambridge, MA, pp. 605–611.

Irani, M., Rousso, B. & Peleg, S. (1994*a*), 'Computing occluding and transparent motions', *Intl. Journal of Computer Vision* **12**(1), 5–16.

Irani, M., Rousso, B. & Peleg, S. (1994*b*), Recovery of ego-motion using image stabilization, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Seattle, WA, pp. 454–460.

Ishikawa, H. & Geiger, D. (1998), Occlusions, discontinuities, and epipolar lines in stereo, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Freiburg, Germany, pp. 232–248.

Ishikawa, H. & Geiger, D. (1999), Mapping image restoration to a graph problem, *in* 'Proc. IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing', pp. 20–23.

Itti, L., Koch, C. & Niebur, E. (1998), 'A model of saliency-based visual attention for rapid scene analysis', *IEEE Trans. Pattern Analysis and Machine Intelligence* **20**(11), 1254–1259.

Jepson, A. & Heeger, D. (1990), Subspace methods for recovering rigid motion II: Theory, Technical Report RBCV-TR-90-36, University of Toronto.

Jepson, A. & Heeger, D. (1992), Linear subspace methods for recovering translation direction, Technical Report RBCV-TR-92-40, University of Toronto, Dept. of Computer Science.

Jones, D. & Malik, J. (1992), 'Computational framework for determining stereo correspondence from a set of linear spatial filters', *Image and Vision Computing* **10**, 699–708.

Ju, S., Black, M. & Jepson, A. (1996), Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Francisco, CA, pp. 307–314.

Julesz, B. (1971), *Foundations of Cyclopean Perception*, University of Chicago Press, Chicago, IL.

Kanade, T. & Okutomi, M. (1994), 'A stereo matching algorithm with an adaptive window: Theory and experiment', *IEEE Trans. Pattern Analysis and Machine Intelligence* **16**(9), 920–932.

Kanatani, K. (1993*a*), '3-d interpretation of optical flow by renormalization', *Intl. Journal of Computer Vision* **11**(3), 267–282.

Kanatani, K. (1993*b*), Renormalization for unbiased estimation, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Berlin, Germany, pp. 599–606.

Koch, C. & Ullman, S. (1985), 'Shifts in selective visual attention : towards the underlying neural circuitry', *Human neurobiology* **4**, 219–227.

Koenderink, J. & van Doorn, A. (1975), 'Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer', *Optica Acta* **22**(9), 773–791.

Konolige, K. (1997), Small vision systems: Hardware and implmentation, *in* 'Proc. Intl. Symposium on Robotics Research', Salt Lake City, UT, pp. 203–212.

Lai, S. & Vemuri, B. (1998), 'Reliable and efficient computation of optical flow', *Intl. Journal of Computer Vision* **29**(2), 87–105.

Liu, H., Hong, T., Herman, M. & Chellappa, R. (1997), 'A general motion model and spatiotemporal filters for computing optical-flow', *Intl. Journal of Computer Vision* **22**(2), 141–172.

Longuet-Higgins, H. (1981), 'A computer algorithm for reconstructing a scene from two projections', *Nature* **293**, 133–135.

Longuet-Higgins, H. & Prazdny, K. (1980), 'The interpretation of a moving retinal image', *Philosophical Transactions of Royal Society of London* **B 208**, 385–397.

Lucas, B. & Kanade, T. (1981), An iterative image registration technique with an application to stereo vision, *in* 'Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Vancouver, Canada, pp. 674–679.

Ma, Y., Kosecka, J. & Sastry, S. (2000), 'Linear differential algorithm for motion recovery: A geometric approach', *Intl. Journal of Computer Vision* **36**(1), 71–89.

Ma, Y., Kosecka, J. & Sastry, S. (2001), 'Optimization criteria and geometric algorithms for motion and structure estimation', *Intl. Journal of Computer Vision* **44**(3), 219–249.

MacLean, W. (1999), Removal of translation bias when using subspace methods, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Corfu, Greece, pp. 753–758.

Maes, F., Collignon, A., Vandermeulen, D., Marchal, G. & Suetens, P. (1997), 'Multi-modality image registration by maximization of mutual information', *IEEE Transactions on Medical Imaging* **16**(3), 187–198.

Maki, A., Bretzner, L. & Eklundh, J. (1995), Local fourier phase and disparity estimation: An analytical study, *in* 'Proc. Intl. Conf. Computer Analysis of Images and Patterns', pp. 868–874.

Maki, A., Nordlund, P. & Eklundh, J.-O. (2000), 'Attentional scene segmentation: Integrating depth and motion', *Computer Vision and Image Understanding* **78**(3), 351–373.

Marr, D. (1982), *Vision: A computational investigation into the human representation and processing of visual information*, W. H. Freeman, San Fransisco.

Marr, D. & Poggio, T. (1976), 'Cooperative computation of stereo disparity', *Science* **194**, 283–287.

Marr, D. & Poggio, T. (1979), 'A computational theory of human stereo vision', *Proceedings of the Royal Society* **B 204**, 301–328.

Maybank, S. (1987), A theoretical study of optical flow, PhD thesis, University of London.

Mayhew, J. & Frisby, J. (1981), 'Psychophysical and computational studies towards a theory of human stereopsis', *Artificial Intelligence* **17**, 349–385.

Memin, E. & Perez, P. (2002), 'Hierarchical estimation and segmentation of dense motion fields', *Intl. Journal of Computer Vision* **46**(2), 129–155.

Milanese, R., Wechsler, H., Gil, S., Bost, J. & Pun, T. (1994), Integration of bottom-up and top-down cues for visual attention using non-linear relaxation, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Seattle, WA, pp. 781–785.

Moravec, H. (1977), Towards automatic visual obstacle avoidance, *in* 'Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Cambridge, MA, p. 584.

Moravec, H. (1981), Rover visual obstacle avoidance, *in* 'Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)', Vancouver, Canada, pp. 785–790.

Morimoto, C. & Chellappa, R. (1996), Fast electronic digital image stabilization, *in* 'Proc. Intl. Conf. on Pattern Recognition (ICPR)', Vienna, Austria, pp. 3:284–288.

Morimoto, C. & Chellappa, R. (1997), Evaluation of image stabilization algorithms, *in* 'Proc. DARPA Image Understanding Workshop', New Orleans, LA, pp. 295–302.

Murray, D., Bradshaw, K., McLauchlan, P., Reid, I. & Sharkey, P. (1995), 'Driving saccade to pursuit using image motion', *Intl. Journal of Computer Vision* **16**(3), 205–228.

Nagel, H. & Enkelmann, W. (1986), 'An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences', *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**(5), 565–593.

Nishihara, H. (1984), 'Practical real-time imaging stereo matcher', *Optical Engineering* **23**(5), 536–545.

Nordlund, P. & Eklundh, J.-O. (1999), Real-time maintenance of figure-ground segmentaion, *in* 'Proc. Intl. Conference on Computer Vision Systems', Las Palmas, Gran Canaria, pp. 115–134.

Ohta, Y. & Kanade, T. (1985), 'Stereo by intra- and inter-scanline search using dynamic programming', *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(2), 139–154.

Okutomi, M. & Kanade, T. (1991), A multiple baseline stereo, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Lahaina, Maui, Hawaii, pp. 63–69.

Oliensis, J. (1996), Rigorous bounds for two-frame structure from motion, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Cambridge, UK, pp. II:184–195.

Oliensis, J. (2000), 'A new structure-from-motion ambiguity', *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(7), 685–700.

Oliensis, J. & Genc, Y. (1999), New algorithms for two-frame structure from motion, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Corfu, Greece, pp. 737–744.

Oliensis, J. & Govindu, V. (1999), 'An experimental study of projective structure from motion', *IEEE Trans. Pattern Analysis and Machine Intelligence* **21**(7), 665–671.

Ong, E. & Spann, M. (1999), 'Robust optical flow computation based on least-median-of-squares regression', *Intl. Journal of Computer Vision* **31**(1), 51–82.

Pahlavan, K., Uhlin, T. & Eklundh, J.-O. (1992), 'Integrating primary ocular processes', *Image and Vision Computing* **10**, 645–662.

Pahlavan, K., Uhlin, T. & Eklundh, J.-O. (1993), *Active Vision, Advances in Computer Science*, Lawrence Erlbaum Associates, chapter Active vision as a methodology, pp. 19–46.

Pollard, S., Mayhew, J. & Frisby, J. (1985), 'Pmf: A stereo correspondence algorithm using a disparity gradient limit', *Perception* **14**, 449–470.

Prazdny, K. (1981), 'Determining the instantaneous direction of motion from optical flow generated by a curvilinearly moving observer', *Computer Graphics Image Processing* **17**(3), 238–248.

Prazdny, K. (1983), 'On the information in optical flows', *Computer Vision Graphics and Image Processing* **22**(2), 239–259.

Prazdny, K. (1985), 'Detection of binocular disparities', *Biological Cybernetics* **52**, 93–99.

Press, W. H., Teukolsky, S., Vetterling, W. T. & Flannery, B. P. (1992), *Numerical Recipes in C*, 2nd edn, Cambridge University Press.

Pritchett, P. & Zisserman, A. (1998), Wide baseline stereo matching, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Bombay, India, pp. 754–760.

Quam, L. (1984), Hierarchical warp stereo, *in* 'Proc. DARPA Image Understanding Workshop', New Orleans, LA, pp. 149–155.

Ramachandran, V. S. (1985), 'Apparent motion of subjective surfaces', *Perception* **14**, 127–134.

Rao, R. P. N., Zelinsky, G. J., Hayhoe, M. M. & Ballard, D. H. (1997), Eye movements in visual cognition: A computational study, Technical Report NRL97.1, National Resource Laboratory for the Study of Brain and Behavior, University of Rochester.

Rieger, J. & Lawton, D. (1985), 'Processing differential image motion', *Journal of the Optical Society of America* A **2**, 254–260.

Roy, S. & Cox, I. (1998), A maximum-flow formulation of the N-camera stereo correspondence problem, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Bombay, India, pp. 492–499.

Sanger, T. D. (1988), 'Stereo disparity computation using Gabor filters', *Biological Cybernetics* **59**, 405–418.

Sara, R. (1999), The class of stable matchings for computational stereo, Technical Report CTU-CMP-1999-22, Czech Technical University.

Satoh, K. & Ohta, Y. (1996), Occlusion detectable stereo: Systematic comparison of detection algorithms, *in* 'Proc. Intl. Conf. on Pattern Recognition (ICPR)', Vienna, Austria, p. A7A.5.

Scharstein, D. & Szeliski, R. (1996), Stereo matching with nonlinear diffusion, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Francisco, CA, pp. 343–350.

Scharstein, D. & Szeliski, R. (2002), 'A taxonomy and evaluation of dense two-frame stereo correspondence algorithms', *Intl. Journal of Computer Vision* **47**(1), 7–42.

Sharkey, P. M., Murray, D. W., Vandevelde, S., Reid, I. D. & McLauchlan, P. F. (1993), 'A modular head/eye platform for real-time reactive vision', *Mechatronics* **3**(4), 517–535.

Simoncelli, E., Adelson, E. & Heeger, D. (1991), Probability distributions of optical flow, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Lahaina, Maui, HI, pp. 310–315.

Singh, A. (1990), An estimation-theoretic framework for image-flow computation, *in* 'Proc. Intl. Conf. on Computer Vision (ICCV)', Osaka, Japan, pp. 168–177.

Smith, S. & Brady, J. (1997), 'SUSAN: A new approach to low-level image-processing', *Intl. Journal of Computer Vision* **23**(1), 45–78.

Soatto, S. & Brockett, R. (1998), Optimal structure from motion: local ambiguities and global estimates, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Santa Barbara, CA, pp. 282–288.

Soatto, S. & Perona, P. (1997), 'Recursive 3-d visual-motion estimation using subspace constraints', *Intl. Journal of Computer Vision* **22**(3), 235–259.

Srinivasan, S. (2000), 'Extracting structure from optical flow using the fast error search technique', *Intl. Journal of Computer Vision* **37**(3), 203–230.

Stiller, C. & Konrad, J. (1999), 'Estimating motion in image sequences', *IEEE Signal Processing Magazine* **16**, 70–91.

Szeliski, R. & Kang, S. (1997), 'Shape ambiguities in structure-from-motion', *IEEE Trans. Pattern Analysis and Machine Intelligence* **19**(5), 506–512.

Szeliski, R. & Shum, H. (1996), 'Motion estimation with quadtree splines', *IEEE Trans. Pattern Analysis and Machine Intelligence* **18**(12), 1199–1210.

Szeliski, R. & Zabih, R. (1999), An experimental comparison of stereo algorithms, *in* 'Proc. Vision Algorithms: Theory and Practice', Corfu, Greece, pp. 1–19.

Tell, D. (2002), Wide baseline matching with applications to visual servoing, PhD thesis, Department of Numerical Analysis and Computer Science, Royal Institute of Technology.

Thomas, I., Simoncelli, E. & Bajcsy, R. (1994), Linear structure from motion, Technical Report MS-CIS-94-61, University of Pennsylvania.

Thompson, W. (1998), 'Exploiting discontinuities in optical flow', *Intl. Journal of Computer Vision* **30**(3), 163–173.

Thompson, W., Mutch, K. & Berzins, V. (1985), 'Dynamic occlusion analysis in optical flow fields', *IEEE Trans. Pattern Analysis and Machine Intelligence* **7**(4), 374–383.

Tian, Q. & Huhns, M. (1986), 'Algorithms for subpixel registration', *Computer Vision Graphics and Image Processing* **35**(2), 220–233.

Tian, T., Tomasi, C. & Heeger, D. (1996), Comparison of approaches to egomotion computation, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Francisco, CA, pp. 315–320.

Tomasi, C. & Shi, J. (1993), Direction of heading from image deformations, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', New York, NY, pp. 422–427.

Torr, P. (1995), Motion Segmentation and Outlier Detection, PhD thesis, Department of Engineering Science, University of Oxford.

Torr, P. & Murray, D. (1997), 'The development and comparison of robust methods for estimating the fundamental matrix', *Intl. Journal of Computer Vision* **24**(3), 271–300.

Toscani, G. & Faugeras, O. (1986), Structure and motion from two noisy perspective images, *in* 'Proc. IEEE Conf. on Robotics and Automation', San Francisco, CA, pp. 221–227.

Toyama, K. & Hager, G. (1999), 'Incremental focus of attention for robust vision-based tracking', *Intl. Journal of Computer Vision* **35**(1), 45–63.

Treisman, A. & Gelade, G. (1980), 'A feature-integration theory of attention', *Cognitive Psychology* **12**, 97–136.

Treisman, A. & Sato, S. (1990), 'Conjunction search revisited', *Journal of Experimental Psychology: Human Perception and Performance* **16**, 459–478.

Tretiak, O. & Pastor, L. (1984), Velocity estimation from image sequences with second order differential operators, *in* 'Proc. Intl. Conf. on Pattern Recognition (ICPR)', Montreal, Canada, pp. 16–19.

Tsai, R. & Huang, T. (1981), 'Estimating 3-d motion parameters of a rigid planar patch', *IEEE Trans. on Acoustic, Speech and Signal Processing* **29**(6), 1147–1152.

Tsai, R. & Huang, T. (1984), 'Uniqueness and estimation of 3-d motion parameters and rigid bodies with curves surfaces', *IEEE Trans. Pattern Analysis and Machine Intelligence* **6**(1), 13–27.

Uras, S., Girosi, F., Verri, A. & Torre, V. (1988), 'A computational approach to motion perception', *Biological Cybernetics* **60**, 79–87.

von Helmholtz, H. (1925), *Treatise on physiological optics*, Dover Publications, New York. Translation: J. P. C. Southall.

Wang, J. & Adelson, E. (1993), Layered representation for motion analysis, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', New York, NY, pp. 361–366.

Wang, J. & Adelson, E. (1994), 'Representing moving images with layers', *IEEE Trans. Image Processing* **3**(5), 625–638.

Weber, J. & Malik, J. (1995), 'Robust computation of optical-flow in a multiscale differential framework', *Intl. Journal of Computer Vision* **14**(1), 67–81.

Weber, J. & Malik, J. (1997), 'Rigid-body segmentation and shape-description from dense optical-flow under weak perspective', *IEEE Trans. Pattern Analysis and Machine Intelligence* **19**(2), 139–143.

Weiss, Y. & Adelson, E. (1996), A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Francisco, CA, pp. 321–326.

Weng, J., Ahuja, N. & Huang, T. (1993), 'Optimal motion and structure estimation', *IEEE Trans. Pattern Analysis and Machine Intelligence* **15**(9), 864–884.

Wolfe, J., Cave, K. & Franzel, S. (1989), 'Guided search: an alternative to the feature integration model of visual search', *Journal of Experimental Psychology: Human Perception and Performance* **15**, 419–433.

Yao, Y. & Chellappa, R. (1996), Selective stabilization of images acquired by unmanned ground vehicles, *in* 'Proc. Intl. Conf. on Pattern Recognition (ICPR)', Vienna, Austria, p. C74.4.

Yau, W. & Wang, H. (1999), 'Fast relative depth computation for an active stereo vision system', *Real-Time Imaging* **5**(3), 189–202.

Young, G. & Chellappa, R. (1992), 'Statistical analysis of inherent ambiguities in recovering 3-d motion from a noisy flow field', *IEEE Trans. Pattern Analysis and Machine Intelligence* **14**(10), 995–1013.

Zabih, R. & Woodfill, J. (1994), Non-parametric local transforms for computing visual correspondence, *in* 'Proc. European Conf. on Computer Vision (ECCV)', Stockholm, Sweden, pp. 151–158.

Zelnik-Manor, L. & Irani, M. (2000), 'Multi-frame estimation of planar motion', *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(10), 1105–1116.

Zhang, T. & Tomasi, C. (1999), Fast, robust, and consistent camera motion estimation, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', Fort Collins, CO, pp. I:164–170.

Zhang, Z., Deriche, R., Faugeras, O. & Luong, Q. (1995), 'A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry', *Artificial Intelligence* **78**(1-2), 87–119.

Zheng, Q. & Chellappa, R. (1993), 'A computational vision approach to image registration', *IEEE Trans. Image Processing* **2**(3), 311–326.

Zhuang, X., T.S., H. & Ahuja, N. (1988), 'A simplified linear optic flow motion algorithm', *Computer Vision Graphics and Image Processing* **42**, 334–344.

Zitnick, C. & Kanade, T. (2000), 'A cooperative algorithm for stereo matching and occlusion detection', *IEEE Trans. Pattern Analysis and Machine Intelligence* **22**(7), 675–684.

Zoghiami, I., Faugeras, O. & Deriche, R. (1997), Using geometric corners to build a 2d mosaic from a set of images, *in* 'Proc. IEEE Computer Vision and Pattern Recognition (CVPR)', San Juan, PR, pp. 420–425.