Abstract


A multiresolution algorithm for energy-based estimation and representation of local spatiotemporal structure by second order symmetric tensors is presented.

The problem of how to properly process estimates with varying degree of reliability is addressed. An efficient spatiotemporal implementation of a certainty-based signal modelling method called normalized convolution is described.

As an application of the above results, a smooth pursuit motion tracking algorithm that uses observations of both target motion and position for camera head control and motion prediction is described. The target is detected using a novel motion field segmentation algorithm which assumes that the motion fields of the target and its immediate vicinity, at least occasionally, each can be modelled by a single parameterised motion model. A method to eliminate camera-induced background motion in the case of a pan/tilt rotating camera is suggested.
I dedicate this thesis to my teachers whose encouragement and stimulation have been invaluable to me.
Acknowledgements

First of all, I wish to thank past and present colleagues at the Computer Vision Laboratory for being great friends. It is a privilege to work with such talented people.

I thank Professor Gösta Granlund, head of the Laboratory, for showing confidence in me and my work.

Professor Hans Knutsson and Drs. Mats T. Andersson, Carl–Fredrik Westin and Carl–Johan Westelius have contributed with many ideas and suggestions. I am particularly grateful to Dr. Westelius, who developed the advanced graphics simulation (‘virtual reality’) that I have used extensively in my work.

I thank Professors Henrik I. Christensen, Erik Granum and Henning Nielsen, and Steen Kristensen and Paolo Pirjanian, all at Laboratory of Image Analysis, University of Aalborg, Denmark, for their kind hospitality during my visits.

Dr. Klas Nordberg and Morgan Ulvklo significantly reduced the number of errors in the manuscript.

Johan Wiklund provided excellent technical support and help with computer related problems.

This work was done within the European Union ESPRIT–BRA 7108 Vision as Process (VAP–II) project. Financial support from the Swedish National Board for Industrial and Technical Development (NUTEK) and the Swedish Defence Materials Administration (FMV) is gratefully acknowledged.
“Free fall is the natural state of motion.”

- In C. W. Misner, K. S. Thorne, J. A. Wheeler: *Gravitation*
# Contents

1 INTRODUCTION .................................................. 1
  1.1 Motivation .................................................. 1
  1.2 The thesis ................................................... 1
  1.3 Notation ...................................................... 2
  1.4 Representing local structure with tensors ................. 3
    1.4.1 Energy-based local structure representation and estimation .................................................. 3
    1.4.2 Efficient implementation of spherically separable quadrature filters ......................... 9

2 MULTiresolution Sparse TENSOR FIELD ..................... 11
  2.1 Low-pass filtering, subsampling, and energy distribution ........ .................. 11
  2.2 Building a consistent pyramid ................................ 14
  2.3 Experimental results ........................................ 21
  2.4 Computation of dense optical flow .......................... 25

3 A SIGNAL CERTainty APPROACH TO SpatiotEMPORAL FILTERING AND MODELLING ............. 31
  3.1 Background .................................................. 31
  3.2 Normalized and differential convolution ......................... 33
  3.3 NDC for spatiotemporal filtering and modelling .................. 36
  3.4 Experimental results ......................................... 41
  3.5 Discussion ................................................... 41

4 MOTION SEGMENTATION AND TRACKING ....................... 49
  4.1 Space–variant sensing ........................................ 50
  4.2 Control aspects ............................................... 50
  4.3 Motion models and segmentation ................................. 54
    4.3.1 Segmentation .............................................. 54
    4.3.2 Discussion ................................................. 58
    4.3.3 Motion models ............................................. 59
  4.4 On the use of normalized convolution in tracking .......... 63
  4.5 Experimental results ......................................... 63

5 SUMMARY ......................................................... 71
A TECHNICAL DETAILS

A.1 Local motion and tensors .................................................. 73
   A.1.1 Velocity from the tensor ........................................... 73
   A.1.2 Speed and eigenvectors ............................................. 74
   A.1.3 Speed and $T_{xy}$ .................................................... 74
A.2 Motion models ............................................................. 76
   A.2.1 Accuracy of affine parameter estimates ....................... 76
This first chapter is intended to give [most of] the necessary background information needed to understand the new results presented in the later chapters.

1.1 Motivation

It is not particularly difficult to convince anyone working with image processing that efficient algorithms for motion estimation are of central importance to the field. Areas such as video coding and real–time robot vision are highly dependent on accurate and fast methods to extract spatiotemporal signal features for segmentation and response generation.

The work presented in this thesis was prompted by certain breakthroughs in the implementation of spatiotemporal filter banks, which seemed to allow completely new applications of massive spatiotemporal filtering and modelling. Could it be that highly accurate motion estimation not only gives a quality improvement to standard real–time algorithms, but actually makes completely new, more powerful methods realizable?

We cannot claim to have a definite answer to whether high performance spatiotemporal filtering is useful in applications with time–constraints—there are some very promising results based on more qualitative aspects of motion fields that seem not to require massive 3D computation—but it is clear that as computers get faster, more powerful methods than those that have been used up to now will inevitably find their way to real–time applications.

1.2 The thesis

Section 1.4 of this chapter is an introduction to energy–based approaches to spatiotemporal local structure estimation and representation. It also provides an outline of certain recent technical results that motivated the work presented in the thesis.
In Chapter 2 we present a novel efficient multiresolution algorithm for accurate and reliable estimation and representation of local spatiotemporal structure over a wide range of image displacements. First we describe how the local spatiotemporal spectral energy distribution is affected by spatial lowpass filtering and subsampling, and we investigate the practical consequences of this. Then we consider how to process a resolution pyramid of local structure estimates so that it contains as accurate and reliable estimates as possible—we provide experimental results that support our reasoning.

In Chapter 3 the concept of signal certainty–based filtering and modelling is treated. This addresses the important problem of how to properly process estimates with varying degree of reliability. The chapter contains a general account of a method called normalised convolution. We look at how this can be efficiently implemented in the case of spatiotemporal filtering and modelling, and present experimental results that demonstrate the power of the method.

Chapter 4 describes an active vision application of the concepts developed in the earlier chapters. We present a smooth pursuit motion tracking algorithm that uses observations of both target position and velocity. The chapter contains novel efficient motion segmentation algorithms for extraction of the target motion field. Excerpts from a number of simulated and real tracking sequences are provided.

1.3 Notation

In choosing the various symbols, certain general principles have been followed. Vectors are denoted by bold–faced lower–case characters, e.g., \( \mathbf{u} \), and tensors of higher order by bold-faced upper–case characters, e.g., \( \mathbf{I} \). The norm of a tensor \( \mathbf{A} \) is denoted by \( \| \mathbf{A} \| \). This will usually mean the Frobenius norm. A ‘hat’ above a vector indicates unit norm, e.g., \( \hat{\mathbf{u}} \). Eigenvalues are denoted by the letter \( \lambda \), and are ordered \( \lambda_1 \geq \lambda_2 \ldots \).

A certain abuse of notation will be noted, in that to simplify notation we do not discriminate between tensors and their coordinate matrices that always turn up in numerical calculations, e.g., \( \mathbf{A}^T \) will denote the transpose of the matrix \( \mathbf{A} \).

Occasionally we will use the notation \( < \mathbf{a}, \mathbf{b} > \) for the inner product between two possibly complex–valued vectors. This is equivalent to the coordinate vector operation \( \mathbf{a^*b} \), where the star denotes complex conjugation and transposition.

The signal is usually called \( f(x) \). This always refers to a local coordinate system centred in a local neighbourhood. The Fourier transform of a function \( s(x) \) is denoted by \( S(u) \). The Laplace transform of a function \( g(t) \) is denoted by \( G(s) \).
1.4 Representing local structure with tensors

This section gives the necessary background information concerning the estimation and representation of local spatio-temporal structure. The concepts presented here, and the fact that the estimation procedure can be efficiently implemented, form a prerequisite for the methods presented in later chapters.

The inertia tensor method is by Bigün, [16, 17, 15, 18]. Pioneer work on its application to motion analysis was done by Jähne, [53, 54, 55, 56]. The local structure tensor representation and estimation by quadrature filtering was conceived by Knutsson, [59, 60, 61, 62, 66, 65]. A comprehensive description of the theory and its application is given in [43]. The efficient implementation of multidimensional quadrature filtering is by Knutsson and Andersson, [63, 64, 6].

1.4.1 Energy-based local structure representation and estimation

One of the principal goals of low-level computer vision can be formulated as the detection and representation of local anisotropy. Typical representatives are lines, edges, corners, crossings and junctions. The reason why these features are important is that they are of a generic character yet specific enough to greatly facilitate unambiguous image segmentation. In motion analysis we are interested in features that are stable in time so that local velocity may be computed from the correspondence of features in successive frames. Indeed, many motion estimation algorithms work by matching features from one frame to the next. A problem with this strategy is that there may be multiple matches, particularly in textured regions. Assuming that a feature is stable between several successive frames, the local velocity may be found from a spatio-temporal vector pointing in the direction of least signal variation\(^1\). The Fourier transform\(^2\) of a signal that is constant in one direction is confined to a plane through the origin perpendicular to this direction, Figure 1.2. If the feature is also constant in a spatial direction, i.e., it is an edge or a line, there will be two orthogonal spatio-temporal directions with small signal variation. This prevents finding the true motion direction, but constrains it to lie in a plane orthogonal to the direction of maximum signal variation. A signal that varies in a single direction has a spectrum confined to a line in this direction through the origin in the Fourier domain, Figure 1.1. The discussion has focused the attention on the angular

\(^1\)There are several ways to more exactly define the directional signal variation, leading to different algorithms. For our present discussion an intuitive picture is sufficient.

\(^2\)When referring to the Fourier transform of the signal we always mean the transform of a windowed signal, a 'local Fourier transform'.
distribution of the signal spectrum and its relation to motion estimation. We will here review two related methods to obtain a representation of this distribution.

In the first method one estimates the inertia tensor $\mathbf{J}$ of the power spectrum, in analogy with the inertia tensor of mechanics. The inertia tensor comes from finding the direction $\hat{n}_{\text{min}}$ that minimises

$$\mathcal{J}[\hat{n}] = \int \left[ \left| \mathbf{u} - (\mathbf{u}^T \hat{n}) \hat{n} \right|^2 \mathcal{F}(\mathbf{u}) \right] d\mathbf{u}$$

which is simply the energy-weighted integral of the orthogonal distances of the data points to a line through the origin in the $\hat{n}$-direction. The motivation for introducing $\mathcal{J}$ is that for signals with a dominant direction of variation, this direction is a global minimum of $\mathcal{J}$. Likewise, we see that for a signal that is constant in one unique direction, this direction is a global maximum of $\mathcal{J}$. A little algebra leads to the following equivalent expression

$$\mathcal{J}[\hat{n}] = \hat{n}^T \mathbf{J} \hat{n}$$

In component form the inertia tensor is given by

$$J_{jk} = \int \left[ \mathcal{F}(\mathbf{u}) \right] \left[ (\mathbf{u}^T \hat{n}) \hat{n} \right] \delta_{jk} - u_j u_k \right] d\mathbf{u}$$

Since $\mathbf{J}$ is a symmetric second order tensor it is now apparent that $\mathcal{J}$ is minimised by the eigenvector of $\mathbf{J}$ corresponding to the smallest eigenvalue. From Plancherel’s relation and using the fact that multiplication by $iu_k$ in the Fourier domain corresponds to partial derivation $\frac{\partial}{\partial x_k}$ in the spatial domain, Equation (1.1) may be written

$$J_{jk} = \int \left[ \left| \nabla \mathcal{F} \right|^2 \delta_{jk} - \frac{\partial \mathcal{F}}{\partial x_j} \frac{\partial \mathcal{F}}{\partial x_k} \right] d\mathbf{x}$$

The integration is taken over the window defining the local spatio-temporal neighbourhood and may be organised as low-pass filtering of the derivative product volumes. The derivatives themselves can be efficiently computed with one-dimensional filters.

Critics of the inertia tensor approach argue that the spatio-temporal low-pass filtering causes an unwanted smearing, and that the use of squared derivatives demands a higher sampling rate than is needed to represent the original signal.

An alternative approach to energy-based local structure estimation and representation was given by Knutsson. This procedure also leads to a symmetric second-order tensor representation but it avoids the spatio-temporal averaging of the inertia tensor approach,\footnote{This problem is particularly severe at motion boundaries and for small objects moving against a textured background.}
1.4 REPRESENTING LOCAL STRUCTURE WITH TENSORS

Spatial domain

Fourier domain

Figure 1.1: A planar autocorrelation function in the spatial domain corresponds to energy being distributed on a line in the Fourier domain. (Iso-surface plots).

Figure 1.2: An autocorrelation function concentrated on a line in the spatial domain corresponds to a planar energy distribution in the Fourier domain. (Iso-surface plots).

Figure 1.3: A spherical autocorrelation function in the spatial domain corresponds to a spherical energy distribution in the Fourier domain. (Iso-surface plots).
and the shift of the spectrum to higher frequencies. The concept of the local structure tensor comes from the observation that an unambiguous representation of signal orientation is given by

\[ T = A\hat{x}\hat{x}^T \]  

where \( \hat{x} \) is a unit vector in the direction of maximum signal variation and \( A \) is any scalar greater than zero. Estimation of \( T \) is done by probing Fourier space in several directions \( \hat{n}_k \) with filters that each pick up energy in an angular sector centred at a particular direction. These filters are complex-valued quadrature filters, which means that their real and imaginary parts are reciprocal Hilbert transforms[22]. The essential result of this is that the impulse response of the filter has a Fourier transform that is real and non-zero only in a half-space \( \hat{n}_k^T \mathbf{u} > 0 \). One designs the quadrature filters to be spherically separable in the Fourier domain, which means that they can be written as a product of a function of radius and a function of direction

\[ F_k(u) = R(p)D_k(\hat{u}) \]

The radial part, \( R(p) \), is made positive in a pass-band so that the filter response corresponds to a weighted average of the spectral coefficients in a region of Fourier space\(^5\). Figure 1.4 shows the radial part of the filter used in most experiments in this thesis. This is a lognormal frequency function, which means that it is gaussian on a logarithmic scale. The real benefit from using quadrature filters comes from taking the absolute value (magnitude) of the filter response. This is a measure of the ‘size’ of the average spectral coefficient in a region and is for narrow–banded signals invariant to shifts of the signal, which then assures that the orientation estimate is independent of whether it is obtained at an edge or on a line (phase invariance). The filter response magnitudes \( |q_k| \) are used as coefficients in a linear summation of basis tensors

\[ T_{est} = \sum_k |q_k| \mathbf{M}^k \]  

where \( \mathbf{M}^k \) is the dual of the outer product tensor \( \mathbf{N}_k = \hat{n}_k\hat{n}_k^T \). The \( \mathbf{M}^k \)'s are defined by the reconstruction relation

\[ \sum_k (\mathbf{S} \cdot \mathbf{N}_k) \mathbf{M}^k = \mathbf{S} \quad \text{for all second order symmetric tensors } \mathbf{S} \]  

where

\[ \mathbf{A} \cdot \mathbf{B} = \sum_{ij} A_{ij}B_{ij} \]

\(^4\)Early approaches to motion analysis by spatio-temporal quadrature filtering are by Adelson and Bergen [1] and Heeger [47]. These papers also discuss the relation to psychophysics and neurophysiology of motion computation.

\(^5\)Note that since the Fourier transform of a real signal is Hermitian there is no loss of information from looking at the spectrum in the filter’s ‘positive direction’ only.
1.4 REPRESENTING LOCAL STRUCTURE WITH TENSORS

defines an inner product in the tensor space. For the reconstruction relation to be satisfied we need at least as many linearly independent filter directions as there are degrees of freedom in a tensor, i.e., six when filtering spatio-temporal sequences. The explicit mathematical expression for the $M^k$'s depends on the angular distribution of the filters, but it is not too difficult to show that the general dual basis element is given by

$$M^k = \sum_i N_i \otimes N_i^{-1} N_k$$

where $\otimes$ denotes a tensor (outer-) product. When the filters are symmetrically distributed in a hemisphere this reduces to

$$M^k = \frac{4}{3} N_k - \frac{1}{4} I$$

where $I$ refers to the identity tensor with components $I_{ij} = \delta_{ij}$.

As discussed earlier in this section, a signal that varies in a single direction $\hat{x}$ (such as a moving edge), referred to as a simple signal, has a spectrum that is non-zero only on a line through the origin in this direction. Writing this as $S(u) = G(\hat{x}^T u) \delta^{line}_{\hat{x}}(u)$, where $\delta^{line}_{\hat{x}}(u)$ is an impulse line in the $\hat{x}$-direction,

6 This is defined as a product of one-dimensional delta-functions in the directions orthogonal to $\hat{x}$.
with a spherically separable quadrature filter can be written

\[
q_k = \int F_k(u) S(u) \, du = \int R(p) D_k(\hat{u}) G(\hat{x}^T u) \Delta_k^{line}(u) \, du \\
= D_k(\hat{x}) \int_0^\infty R(p) G(p) \, dp + D_k(-\hat{x}) \int_0^\infty R(p) G(-p) \, dp \\
= aD_k(\hat{x}) + a^* D_k(-\hat{x})
\]

where \( a \) is a complex number that only depends on the radial overlap between the filter and the signal (\( a^* \) being its complex conjugate). Since the filter is non-zero in only one of the directions \( \pm \hat{x} \), the magnitude of the filter response is

\[
|q_k| = |a| |D_k(\hat{x}) + D_k(-\hat{x})|
\]

Now, if we choose the directional function to be

\[
D_k(\hat{u}) = \begin{cases} 
(\hat{n}_k^T \hat{u})^2 = (\hat{n}_k \hat{n}_k^T) \cdot (\hat{u} \hat{u}^T) & \hat{n}_k^T \hat{u} > 0 \\
0 & \text{otherwise}
\end{cases}
\]

we see that Equation (1.4) reduces to

\[
T_{est} = \sum_k |q_k| M_k = \sum_k |a| (\hat{n}_k \hat{n}_k^T) \cdot (\hat{x} \hat{x}^T) \cdot M_k = \sum_k |a| (N_k \cdot \hat{x} \hat{x}^T) \cdot M_k
\]

This proves that the suggested method correctly estimates the orientation tensor \( T \) of Equation (1.3) in the ideal case of a simple signal. In general we may represent the orientation of a signal by the outer product tensor that minimises the Frobenius distance

\[
\Delta = ||T_{est} - A\hat{x}\hat{x}^T||_F = \sqrt{\sum_{ij}(T_{ij} - A_{ij}x_j)^2}
\]

It is not difficult to see that the minimum value is obtained when \( A \) is taken as the largest eigenvalue of \( T_{est} \) and \( \hat{x} \) the corresponding eigenvector.

Another relevant ideal case besides that of a simple signal is the moving point. The spectrum of such a signal is constant on a plane through the origin and zero outside it. In this case the estimated tensor will have two (equal) non-zero eigenvalues and the eigenvector corresponding to the third (zero-valued) eigenvalue will point in the direction of spatio-temporal motion. In general, wherever there is more than one spatial orientation, the correct image velocity can be recovered from the direction of the eigenvector corresponding to the smallest eigenvalue.

The details of how the velocity components are recovered from the tensor are given in Appendix A.1.1.
1.4.2 Efficient implementation of spherically separable quadrature filters

We saw in the previous subsection that the inertia tensor method allows an efficient implementation using one-dimensional derivative and low-pass filters. We now seek a corresponding efficient implementation of the quadrature filtering method. A problem with the quadrature filters is that spherical separability in general is incompatible with Cartesian separability. We therefore inevitably introduce errors when trying to approximate the multi-dimensional filters by a convolution product of low-dimensional filters. The success of such an attempt naturally depends on the magnitude of the approximation error. Unfortunately, it is not entirely obvious how the filters should be decomposed to simultaneously satisfy the constraints of minimum approximation error and smallest possible number of coefficients. Given the shape of the present filters, with their rather sharp magnitude peak in the central direction of the filter, it is natural to try a decomposition with a one-dimensional quadrature filter in the principal direction and real-valued one-dimensional low-pass filters in the orthogonal directions. With a symmetric distribution of filters, taking full advantage of symmetry properties, this leads to the scheme of Figure 1.5. The filter coefficients are determined in a recursive optimisation process in Fourier space. With an ideal multi-dimensional target filter \( \hat{F}(u) \), and our current

Figure 1.5: 3D sequential quadrature filtering structure.
component product\textsuperscript{7} approximation of this, $F(u) = \prod_k F_k(u)$, we define temporary targets

$$F_i(u) = \frac{\tilde{F}(u)}{\prod_k F_k(u)}$$

for the component filters. We use this target filter to produce a (hopefully) better component filter by minimising the weighted square error

$$\mathcal{E}_i = \| W_i(|\rho|) \prod_{k \neq i} F_i(u) [\tilde{F}_i(u) - F_i^{new}(u)] \|^2$$

$$= \| W_i(|\rho|) [\tilde{F}(u) - F_i^{new}(u) \prod_{k \neq i} F_k(u)] \|^2$$

(1.9)

where $W_i(|\rho|)$ is a radial frequency weight, typically $W_i(|\rho|) \approx |\rho|^{-\alpha + \varepsilon}$, see [43]. Equation (1.9) says that on average we will decrease the difference between the ideal filter $\tilde{F}(u)$ and the product $F(u)$. The constraint that the filter coefficients are to be confined to certain chosen spatial coordinate points (in this case ideally on a line\textsuperscript{8}) is implemented by specifying the discrete Fourier transform to be on the form

$$F_i(u) = \sum_{n=1}^{N_k} f_k(\xi_n) \exp(-i\xi_n \cdot u), \quad |\rho| < \pi$$

where the sum is over all allowed spatial coordinate points $\xi_n$. The optimiser then solves a linear system for the least-square sense optimal coefficients $f_k$.

The optimisation procedure (definition of component target function and optimisation of the corresponding filter) is repeated cyclically for all component filters until convergence. For the present filters this takes less than ten iterations. The quality of the optimised filters has been carefully investigated by estimating the orientation of certain test patterns with known orientation at each position using the local structure tensor method. The result is that the precision is almost identical to what is obtained with full multi-dimensional filters, [6]. As an example, a particular set of full 3D filters requires $9 \times 9 \times 9 \times 6 = 8748$ coefficients. A corresponding set of sequential filters with the same performance characteristics requires 273 coefficients. This clearly makes the quadrature filtering approach competitive in relation to the inertia tensor method\textsuperscript{9} which requires partial derivative computations followed by multi-dimensional low-pass filtering of all independent products of the derivatives.

\textsuperscript{7}Convolution in the space-time domain corresponds to multiplication in the frequency domain.
\textsuperscript{8}Some of the filters are for sampling density reasons "weakly" two-dimensional with a tridiagonal form.
\textsuperscript{9}From a computation cost point of view, assuming the same size of the filters, disregarding the fact that the inertia tensor method requires a higher sampling density to obtain comparable spatial resolution.
A unique feature of our approach to motion segmentation is that models are fitted not to local estimates of velocity but to a sparse field of spatiotemporal local structure tensors. In this chapter we present a fast algorithm for generation of a sparse tensor field where estimates are consistent over scales.

### 2.1 Low-pass filtering, subsampling, and energy distribution

As described in Section 1.4.1 there is a direct correspondence between on one hand the spatiotemporal displacement vector of a local spatial feature and on the other hand the local structure tensor. However, to robustly estimate the tensor, so as to to avoid temporal under-sampling and be able to cope with a wide range of velocities, it is necessary to adopt some kind of multiresolution scheme. Though it is possible to conceive various advanced partitionings of the frequency domain into frequency channels by combinations of spatial and temporal subsampling, [44], we opted to compute a simple low-pass pyramid à la Burt [25] for each frame, and not to resample the sequence temporally. The result is a partitioning of the frequency domain as shown in Figure 2.1. Each level in the pyramid is constructed from the previous level by Gaussian low-pass filtering and subsequent subsampling by a factor two. To avoid aliasing when subsampling it is necessary to use a filter that is quite narrow in the frequency domain. As is seen in Figure 2.2, the result is that the energy is substantially reduced in the spatial directions. This is of no consequence in the ideal case of a moving line or point, when the spectral energy is confined to a line or a plane — the orientation is unaffected by the low-pass filtering. However, the situation is quite different in the case of a noisy signal. Referring to Figure 2.3, a sequence of white noise images that is spatially low-pass filtered and
subsampled becomes orientation biased since energy is lost in the spatial directions. A possible remedy for this is to low-pass filter the signal temporally with a filter that is twice as broad as the filter used in the spatial directions [44]. This type of temporal filter can be efficiently implemented as a cascaded recursive filter, [39]. A couple of experiments were carried out to quantitatively determine the influence of isotropic noise on orientation estimates in spatially subsampled sequences. The orientation bias caused by the low-pass filtering\(^1\) and subsampling was measured by computing the average orientation tensor over each sequence and determining the quotient \(l_t/l_{\text{spat}}\), where \(l_t\) refers to the eigenvector in the temporal direction and \(l_{\text{spat}}\) to the average of the eigenvectors in the spatial plane.

In the first experiment the average orientation of initially white noise was computed. The results are shown in Table 2.1. As expected, the average orientation becomes biased when not compensating for the energy anisotropy by temporal low-pass filtering. However, the effect is quite small for moderate spatial low-pass filtering (\(\sigma_{\text{spat}} = \pi/4\)). This is due to the fact that the quadrature filter used is fairly insensitive to high fre-

\(^1\)The low-pass filters were on the form \(F(\omega_x, \omega_y, \omega_t) = \exp[-(\omega_x^2/\sigma_{\text{spat}}^2) + (\omega_y^2/\sigma_{\text{spat}}^2) + (\omega_t^2/\sigma_t^2)]\).
Figure 2.2: Gaussian low-pass filtering (\( \sigma = \pi/4 \)) reduces the signal energy in the spatial directions. The subsequent subsampling moves the maximum frequency down to \( \pi/2 \) (dashed line). There is also some aliasing caused by the non-zero tail above \( \pi/2 \).

frequencies, cf. Figure 1.4. Repeating the low-pass filtering and subsampling twice with \( \sigma_{\text{spat}} = \pi/4 \) with no temporal filtering results in a quotient of 1.04, which indicates that temporal filtering in fact may be unnecessary. This is important in real-time control applications where long delays cause problems.

<table>
<thead>
<tr>
<th>( \sigma_{\text{spat}} )</th>
<th>( \sigma_t = \infty )</th>
<th>( \sigma_t = 2\sigma_{\text{spat}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \pi/4 )</td>
<td>1.03</td>
<td>1.03</td>
</tr>
<tr>
<td>( \pi/8 )</td>
<td>1.19</td>
<td>1.03</td>
</tr>
<tr>
<td>( \pi/16 )</td>
<td>1.57</td>
<td>1.01</td>
</tr>
</tbody>
</table>

Table 2.1: Results of low-pass filtering and spatial subsampling by a factor two of a sequence of white noise. The numbers show the quotient \( k_t/k_{\text{spat}} \) of the average tensor.

In a second experiment we used a synthetic volume with an unambiguous spatiotemporal orientation at each position. The sequence was designed to have a radially sinusoidal variation of grey-levels with a decreasing frequency towards the periphery, Fig-
Figure 2.3: Illustration of low-pass filtering and subsampling of a white noise sequence. Plots show the spectral energy distribution in one spatial direction and the temporal direction. Left: Spatial low-pass filtering reduces the signal energy in the spatial directions. Middle: Spatial subsampling leads to a rescaling of the spectrum. Right: The energy anisotropy is compensated for by a temporal low-pass filtering with a filter that is twice as broad as the corresponding filter in the spatial direction.

Figure 2.4. The volume was corrupted with white noise and subsequently low-pass filtered ($\sigma_{r_{pad}} = \pi/4$) and subsampled. The change in average orientation caused by the noise remained below one degree with the signal-to-noise ratio as low as 0 dB SNR. This is not surprising since the quadrature filter picks up much less energy from a signal with a random phase distribution over frequencies than from a signal with a well-defined phase. The conclusion of this investigation is that with an appropriate choice of radial filter function of the quadrature filter, i.e., one that is comparatively insensitive to high frequencies, there is no reason to worry about orientation bias induced by uncorrelated signal noise. Consequently, no temporal low-pass filtering is necessary.

### 2.2 Building a consistent pyramid

The construction of a low-pass pyramid from each image frame gives a number of separate spatial resolution channels that can be processed in parallel. Consecutive images
are stacked in a temporal buffer which is convolved with quadrature filters. The magnitude of the filter outputs are used in the composition of local structure tensors. The result is a multiresolution pyramid of tensors. At each original image position there are now several tensors, one from each level of the pyramid, describing the local spatiotemporal structure as it appears at each particular scale. The question arises how to handle this type of representation of the local structure. The answer, of course, depends on the intended application. Our intention is to perform segmentation by fitting parameterised models of the spatiotemporal displacement field to estimates in regions of the image. Interpreting the spatiotemporal displacement as the direction of minimal signal variation, it is clear that the information is readily available in the tensor. For efficiency reasons we want to use the sparse multiresolution tensor field as it is, without any data conversion or new data added. The problem of how to handle the tensor field pyramid then reduces to that of deciding which confidence value should be given to each estimate, i.e., how much a particular tensor should be trusted. For computational efficiency it is also desirable to sort out data that does not contain any useful information as early as possible in a chain of operations.

At this point it is appropriate to make the distinction between two entities of fundamental importance. We use the definitions by Horn, [49]. A point on an object in motion traces out a particle path (flow line) in 3D space, the temporal derivative of which is the instantaneous 3D velocity vector of the point. The geometrical projection of the particle path onto the image plane by the camera gives rise to a 2D particle path whose temporal derivative is the projected 2D velocity vector. The 2D motion field is the collection of all such 2D velocity vectors. The image velocity or optical flow is (any) estimate of the 2D motion field based on the spatiotemporal variation of image intensity. Several
investigators [49, 97, 98, 78, 36, 38] have studied the relation between the 2D motion field and the image velocity. The conclusion is that they generally are different, sometimes very much so. A classical example is by Horn, [49]. A smooth sphere with a specular\(^2\) surface rotating under constant illumination generates no spatiotemporal image intensity variation. On the other hand, if the sphere is fixed but a light source is in motion, there will be a spatiotemporal image intensity variation caused by reflection in the surface of the sphere. The intensity variation caused by this type of “illusory” motion can not be distinguished from that caused by objects in actual motion without a priori knowledge or high-level scene interpretation. A diffusely reflecting (Lambertian) surface also induces intensity variation caused by changes in angle between the surface normal and light sources. This variation is typically a smooth function of position, and independent of texture and surface markings. The conclusion is that the optical flow accurately describes the motion field of predominantly diffusely reflecting surfaces with a large spatial variation of grey level.

The use of the local structure tensor for motion estimation is based on the assumption that the spatiotemporal directions of largest signal variance are orthogonal to the spatiotemporal displacement vector. The shape of the tensor, regarded as an ellipsoid with the semi-axes proportional to the eigenvalues of the tensor, cf. Figures 1.1 – 1.3, is a simple model of the local signal variation, with the longest semi-axis corresponding to the direction of maximum signal variation. It is evident that when the signal variation is close to uniform in all directions, no reliable information about the local motion direction is available. To qualify as a reliable estimate we require the neighbourhood to have a well-defined anisotropy (orientation). This means that the smallest tensor eigenvalue should be substantially smaller than the largest one. It is beneficial to have a computationally cheap measure of anisotropy that does not require eigenvalue decomposition, so that unreliable estimates of velocity may be quickly rejected from further processing. A suitable measure is given by

\[
\mu = \frac{||T||_F}{\text{Tr}(T)} = \frac{\sum_{i,j} T_{ij}^2}{\sum_k T_{kk}} = \sqrt{\frac{\sum_k \lambda_k^2}{\sum_k \lambda_k}}
\]

In Figure 2.5 \(\mu\) is plotted as a function of degree of neighbourhood anisotropy. We simply threshold on \(\mu\), discarding tensors that are not sufficiently anisotropic. The Frobenius norm of the tensor, \(||T||_F\), is a measure of the signal amplitude as seen by the filters. If the amplitude is small, it is possible that the spatial variation of the signal may be too small to dominate over changes in illumination—the tensor becomes very noise sensitive. We therefore reject tensors whose norm is not greater than an energy threshold \(\eta\). To become independent of the absolute level of contrast, one may alternatively reject tensors whose norm is below a small fraction (say, a few percent) of the largest tensor element in a frame.

---

\(^2\)Reflects like a mirror.
Next, consider the relative reliability of tensors at different scales. With the filters (almost) symmetrically distributed in space, there is no significant direction dependence of the angular error in the orientation estimation\(^3\). Consequently we expect the angular error in the estimation of the spatiotemporal displacement vector to be independent of direction. The angular error may be converted into an absolute speed error, which becomes a function of the speed and the scale at which the estimate is obtained, Figure 2.6 (left). Similarly it is interesting to see the angular error at each scale transformed into a corresponding angle at the finest scale, Figure 2.6 (right). This gives an indication of the relative validity of the estimates obtained at different scales as a function of image speed.

There are a couple of additional relevant constraints:

1. The spatial localisation of the estimate should be as good as possible.
2. Temporal aliasing should be avoided.

The first of these demands is of particular importance when using more sophisticated models than a constant translation. It also leads to more accurate results at motion borders. The second item calls for a short digression. Consider a sinusoidal signal of frequency \(\omega_0\) moving with speed \(v_0\) [pixels/frame]. The resulting inter-frame phase shift

\[^3\text{For the test volume of Figure 2.4 corrupted with additive noise, the average magnitude of the angular error is 0.8\(^\circ\), 3.0\(^\circ\), and 9.4\(^\circ\) for a signal–to–noise ratio of \(\approx\)dB, 10dB, and 0dB respectively. [64].}\]
Figure 2.6: Plots of errors as they appear at the finest scale, assuming an angular error $\Delta \phi = 3.0^\circ$. Full line: finest scale, $k=0$. Dotted line: coarsest scale, $k=3$.

Left: Absolute speed error $\Delta v$ as a function of speed $v$ and scale $k$. The functions are $\Delta v(v,k) = 2^k \tan(\arctan(2^{-k}v) + \Delta \phi) - v$.

Right: Apparent angular error $\Delta \phi(v,k)$ as a function of speed $v$ and scale $k$. The functions are $\Delta \phi(v,k) = \arctan(v) - \arctan[2^k \tan(\arctan(2^{-k}v) - \Delta \phi)]$.

is $\omega_0(v_0)$. However, this phase shift is ambiguous, since the signal looks the same if shifted any multiple of the wavelength. In particular, any displacement $v_0$ greater than half the wavelength manifests itself as a phase shift less than $\pi$ (aliasing). One consequence of this is that an unambiguous estimate of spatiotemporal orientation can strictly only be obtained if the speed $v_0$ satisfies

$$v_0 < \frac{\pi}{\omega_0}$$

where $\omega_0$ is the maximum spatial frequency. The combination of a quadrature filter comparatively insensitive to high frequencies and a low-pass filter with a low cut-off frequency reduces the actual high-pass energy influence, so that we dare to extend the maximum allowed velocity estimated at each scale to well above the nominal 1 pixel per frame, particularly at the coarser scales.

The arguments presented above indicate that a tensor validity labelling scheme must include some kind of coarse-to-fine strategy. With initial speed estimates at the coarser scales we can decide whether or not it is useful to descend to finer scales to obtain more precise results. If not, we inhibit the corresponding positions at the finer scales, i.e., we set a flag that indicates that they are invalid, Figure 2.7. The local speed, $s$, is in the moving point case determined by the size of the temporal component of the eigenvector corresponding to the smallest eigenvalue of the tensor. In the case of a single dominant eigenvector (moving edge/line case) the speed is determined by the temporal
2.2 BUILDING A CONSISTENT PYRAMID

Figure 2.7: The use of coarse-to-fine inhibition is here illustrated for a case with a sharp motion gradient, e.g., a motion border between two objects.

component of the eigenvector corresponding to the largest eigenvalue. One finds (cf. Appendix A.1.2),

\[
\begin{cases}
    \sqrt{\frac{c_{11}}{1-c_{11}}} & \text{numerical rank 1 tensor} \\
    \sqrt{\frac{c_{33}}{1-c_{33}}} & \text{numerical rank 2 tensor}
\end{cases}
\]

It appears that we have to compute the eigenvalue decomposition of the tensor. This is actually not the case. The eigenvalues can be efficiently computed as the roots of the (cubic) characteristic polynomial \( \det(\mathbf{T} - \lambda \mathbf{I}) \). Using the fact that a symmetric tensor can always be decomposed into

\[
\mathbf{T} = (\lambda_1 - \lambda_2)\mathbf{T}_1 + (\lambda_2 - \lambda_3)\mathbf{T}_2 + \lambda_3\mathbf{T}_3
\]

with

\[
\mathbf{T}_1 = \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T, \quad \mathbf{T}_2 = \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T, \quad \mathbf{T}_3 = \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T + \hat{\mathbf{e}}_3\hat{\mathbf{e}}_3^T = \mathbf{I}
\]

we create a rank 2 tensor \( \hat{\mathbf{T}} \) by subtracting \( \lambda_3\mathbf{I} \). This new tensor has the same eigenvectors as the original tensor, but eigenvalues \( \hat{\lambda}_1 = \lambda_1 - \lambda_3, \hat{\lambda}_2 = \lambda_2 - \lambda_3 \) and \( \hat{\lambda}_3 = 0 \). Now that the tensor is of (at most) rank 2, there are a couple of interesting relations that we
can use to find the temporal components of the eigenvectors needed to compute the local speed, namely

\[
\text{Tr } \mathbf{T}_{xy} = \hat{\lambda}_1 (1 - e_{13}^2) \quad \text{(numerical rank 1 case)}
\]

\[
\det \mathbf{T}_{xy} = \hat{\lambda}_1 \hat{\lambda}_2 e_{13}^2 \quad \text{(numerical rank 2 case)}
\]

where \( \mathbf{T}_{xy} \) refers to the leading \( 2 \times 2 \)-submatrix of \( \mathbf{T} \), and \( \text{Tr} \) refers to the trace (sum of diagonal elements) of a matrix. See Appendix A.1.3 for a proof of these elementary but perhaps somewhat unobvious relations.

An outline of the algorithm is given in Figure 2.8.
2.3 Experimental results

To demonstrate the method we apply it to a class of sequences, Figure 2.9, representing a typical tracking situation—a small textured surface slowly translating in front of a comparatively rapidly translating background. Each frame is supplemented by a corresponding vector image representing the true displacement field. The investigation focuses on the compatibility between the multi-scale tensor field and the true displacement field. This is done by looking at the deviation $\Delta\phi$ from the anticipated 90° angle between the local spatiotemporal displacement vector and the eigenvector(s) corresponding to the non-zero eigenvalue(s). The appropriate true displacement $\hat{v}$ at a coarse scale is determined by computing the average of the vectors at the four positions of the finer scale that correspond to each position at the coarse scale, followed by a rotation to compensate for

![Figure 2.9: Test sequence. Every tenth frame is shown.](image)
the spatial scale change. Starting at the coarsest scale, at each position we compute a weighted average

\[
<\Delta \phi> = \frac{\tilde{\lambda}_1 \Delta \phi_1 + \tilde{\lambda}_2 \Delta \phi_2}{\lambda_1 + \lambda_2}
\]

where \(\Delta \phi_1\) and \(\Delta \phi_2\) are the deviations of the first and second eigenvectors respectively. The average deviation is then converted into an angle at the next, finer scale\(^4\), and, together with the weight \(\tilde{\lambda}_1 + \tilde{\lambda}_2\), propagated to the corresponding four positions at this scale. A new average deviation angle is computed from the coarse-scale value and the new fine-scale local estimate. Eventually we arrive at the finest scale with an average deviation angle at each position that takes into account estimates at all scales.

Results for two test sequences are shown in Tables 2.3 and 2.4. In the first sequence the background moves at a speed of \(4 \cdot \sqrt{2} \approx 5.7\) pixels/frame, in the second at \(8 \cdot \sqrt{2} \approx 11.3\) pixels/frame. In both cases the foreground texture moves at \(1 \cdot \sqrt{2} \approx 1.4\) pixels/frame. The ‘single scale’ column gives the result when each scale is used alone, covering the entire speed range. The ‘multi-scale w. inhib.’ column refers to the proposed method with multiple scales with each scale covering a certain speed interval.

Following the reasoning in the preceding section, we have a number of parameters to set—the anisotropy threshold \(\mu\), the energy threshold \(\eta\), and (in the multi-scale case) the upper and lower speed limits for each scale. In the experiments we required a density greater than 20 %, i.e., at least every fifth position should have a valid tensor. With this constraint we find that the values of Table 2.2 give accurate results when three levels are used. From Tables 2.3 and 2.4 we see that, as expected, the average error is quite large when only the finest or the coarsest scale is used. The multi-scale method on the other hand, generates quite accurate estimates for both low and high speed regions. In Figure 2.10 an example of the instantaneous angular error as a function of image position is shown.

\(^4\)If \(v\) denotes the speed, the fine-scale angle is given by (cf. Figure 2.6)

\[
\Delta \phi_{\text{fine}} = \arctan(2v) - \arctan(2\tan(arctan(v) - \Delta \phi))
\]
Figure 2.10: Distribution of errors using 1, 2, and 3 levels in the pyramid.
Table 2.3: Average apparent angular error at finest scale in degrees. Background velocity $v_b = (-4, 4)$, foreground velocity $v_f = (1, 1)$. Density $\geq 20\%$.

<table>
<thead>
<tr>
<th>level(s)</th>
<th>single scale</th>
<th>multi-scale w. inhib.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>background</td>
<td>foreground</td>
</tr>
<tr>
<td></td>
<td>background</td>
<td>foreground</td>
</tr>
<tr>
<td>1</td>
<td>11.18</td>
<td>1.73</td>
</tr>
<tr>
<td>2</td>
<td>2.56</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>1.23</td>
<td>5.76</td>
</tr>
</tbody>
</table>

Table 2.4: Average apparent angular error at finest scale in degrees. Background velocity $v_b = (-8, 8)$, foreground velocity $v_f = (1, 1)$. Density $\geq 20\%$.

<table>
<thead>
<tr>
<th>level(s)</th>
<th>single scale</th>
<th>multi-scale w. inhib.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>background</td>
<td>foreground</td>
</tr>
<tr>
<td></td>
<td>background</td>
<td>foreground</td>
</tr>
<tr>
<td>1</td>
<td>10.63</td>
<td>1.91</td>
</tr>
<tr>
<td>2</td>
<td>4.60</td>
<td>1.99</td>
</tr>
<tr>
<td>3</td>
<td>0.86</td>
<td>4.84</td>
</tr>
</tbody>
</table>

ous sequences) were generated. In the first sequence the texture moves at a speed of $1 \cdot \sqrt{2} \approx 1.4$ pixels/frame, in the second at $4 \cdot \sqrt{2} \approx 5.7$ pixels/frame. Figure 2.11 shows histograms of the apparent angular errors. The average errors are $1.26^\circ$ and $0.54^\circ$ respectively. This shows that the present method is capable of producing results which are comparable with any existing motion estimation algorithm.

Finally we provide results for three well-known test sequences used in the investigation by Barron et al. [13]—Fleet’s tree sequences and Quam’s Yosemite sequence\(^5\). The tree sequences, Figure 2.12, were generated by translating a camera with respect to a textured planar surface. In the ‘Translating tree’ sequence the camera moves perpendicular to the line of sight, generating velocities between 1.7 and 2.3 pixels/frame—the surface is tilted. In the ‘Diverging tree’ sequence the camera moves towards the (tilted) surface generating velocities between 0.0 and 2.0 pixels/frame. Note that since the velocity range of the tree sequences is quite narrow, they are not very well suited for study of the performance of multiresolution algorithms. The Yosemite sequence, Figure 2.13, is a computer animation of a flight through Yosemite valley. It was generated by mapping aerial photographs onto a digital terrain map. Motion is predominantly divergent with large local variations due to occlusion and variations in depth. Velocities range from 0 to 5 pixels/frame. The clouds in the background move rightward while changing their shape over time, which makes accurate estimation of their velocity difficult. In Fig-

\(^5\)These and other sequences and their corresponding true flows are available via anonymous ftp from the University of Western Ontario at ftp.csd.uwo.ca/pub/vision/TESTDATA.
2.4 COMPUTATION OF DENSE OPTICAL FLOW

In the present study we found no reason to compute a dense optical flow, since this is a computationally expensive and notoriously ill-posed problem. However, there are situations when a dense field is required, e.g., in region-based velocity segmentation. At least two principally different approaches to computation of dense fields exist. One is to fit parametric models of motion in local image patches, the models ranging from constant motion via affine transformations to mixture models of multiple motions. The fitting is done by clustering, e.g., [85, 57], or [least-squares] optimisation. In Chapter 4 we use

**Figure 2.11**: Histograms of apparent angular error distributions. *Left*: Velocity \( \mathbf{v} = (1.0, 1.0) \). *Right*: \( \mathbf{v} = (4.0, 4.0) \). Density \( \geq 20\% \). Each estimate was given a weight equal to its confidence value.

Figure 2.13 (right), note the larger errors caused by motion discontinuities at the mountain ridges. Numerical results for the three sequences are given in Table 2.5. Figures 2.14 and 2.15 show histograms of the angular errors.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Average error</th>
<th>Standard deviation</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>translating tree</td>
<td>0.30</td>
<td>0.57</td>
<td>66%</td>
</tr>
<tr>
<td>diverging tree</td>
<td>1.72</td>
<td>1.64</td>
<td>47%</td>
</tr>
<tr>
<td>Yosemite</td>
<td>2.41</td>
<td>5.36</td>
<td>48%</td>
</tr>
</tbody>
</table>

**Table 2.5**: Performance for the three test sequences.

2.4 Computation of dense optical flow

In the present study we found no reason to compute a dense optical flow, since this is a computationally expensive and notoriously ill-posed problem. However, there are situations when a dense field is required, e.g., in region-based velocity segmentation. At least two principally different approaches to computation of dense fields exist. One is to fit parametric models of motion in local image patches, the models ranging from constant motion via affine transformations to mixture models of multiple motions. The fitting is done by clustering, e.g., [85, 57], or [least-squares] optimisation. In Chapter 4 we use
**Figure 2.12**: Translating tree. *Left*: Single frame from the sequence. *Right*: Angular error magnitude.

**Figure 2.13**: Yosemite sequence. *Left*: Single frame from the sequence. *Right*: Angular error magnitude.
2.4 COMPUTATION OF DENSE OPTICAL FLOW

Figure 2.14: Tree sequences. Histograms of angular errors. Left: translating tree. Right: diverging tree. Each estimate was given a weight equal to its confidence value.

Figure 2.15: Yosemite sequence. Histogram of angular errors. Each estimate was given a weight equal to its confidence value.

simple parametric model fitting to segment an image into object and background. The second class of methods is based on regularisation theory, e.g., [91, 75], the basic idea of which is to stabilise solutions of ill-posed problems by imposing extra constraints, e.g., spatial smoothness, on the solutions. The problem of computing a dense optical flow is ill-posed since we need two constraints at each position to be able to determine the two velocity components, but we have only one at our disposal at edges or lines, namely the direction of maximum signal variation. We will now discuss the regularisation approach and its relation to the work presented in this chapter.
One way to proceed is to define a cost functional

\[ E = E_1 + \alpha E_2 \]

where the first term is a global measure of the inconsistency between the observed tensor field and the optical flow model, and the second term, \( E_2 \), implements the regularisation by a motion variation penalty. \( \alpha \) is a positive number that determines the relative importance of the two terms. We are looking for the optical flow field that minimises \( E \). Any measure of inconsistency between a tensor and a velocity vector must be based on the fact that the direction of maximum signal variation, coinciding with the largest eigenvector, is orthogonal to the spatiotemporal direction of motion. Whenever the second eigenvalue is significantly greater than the third, smallest one, its corresponding eigenvector is also orthogonal to the direction of motion. A possible inconsistency term is consequently given by

\[
E_1 = \int_{\text{image}} \left[ (\lambda_3 - \lambda_2) (\hat{e}_1 \cdot \tilde{\mathbf{u}})^2 + (\lambda_2 - \lambda_3) (\hat{e}_2 \cdot \tilde{\mathbf{u}})^2 \right] dxdy = \int_{\text{image}} \tilde{\mathbf{u}}^T \hat{T} \tilde{\mathbf{u}} dxdy
\]

where \( \tilde{\mathbf{u}} = (u, v, 1)^T \) is the spatiotemporal displacement vector, and \( \hat{T} \) the 'rank-reduced' tensor introduced in Section 2.2. In the most simple case the regularisation term is an integral [50]

\[
E_2 = \int_{\text{image}} \left[ \left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial u}{\partial y} \right)^2 + \left( \frac{\partial v}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 \right] dxdy
\]

which clearly penalises spatial variation in the motion field—there exist several modifications to avoid unwanted smoothing over velocity discontinuities, e.g., [77, 110]. The minimum of the cost functional is found from the corresponding Euler-Lagrange partial differential equation, e.g., [41], which in this particular case is given by

\[
\hat{\mathbf{t}}_i^T \tilde{\mathbf{u}} - \alpha \Delta u = 0
\]

\[
\hat{\mathbf{t}}_i^T \tilde{\mathbf{u}} - \alpha \Delta v = 0
\]

where \( \hat{\mathbf{t}}_i \) denotes the \( i \)th column vector of \( \hat{T} \), and \( \Delta \) is the Laplacian operator. To efficiently solve this equation one may use some instance of the multi-grid method, [23, 40, 89, 32]. Multi-grid methods take advantage of low-pass pyramid representations in the following way. The equation is converted into a finite difference approximation which is then solved iteratively, e.g., by the Gauss-Seidel or Jacobi methods [42]. When analysing such iterative methods it is found that high-frequency errors are eliminated much more quickly than errors of longer wavelength. These low-frequency errors are however shifted to higher frequencies at a coarser scale where they consequently may be efficiently eliminated. A solution at a coarse scale can, however, not capture fine details and consequently there are high-frequency errors. The idea is then to produce an approximate solution at one level and then use this as an initial value for the iteration at another level where errors in a different frequency band may be efficiently eliminated.

A particularly interesting scheme in the context of multi-level tensor fields is the adaptive
multi-scale coarse-to-fine scheme of Battiti, Amaldi, and Koch. [14]⁶, which may be regarded as a refinement of an old method called block relaxation, where one starts by computing an approximation to the optical flow at the coarsest scale, use an interpolated version of this as an initial value for the iteration at the next finer scale, and so on. The adaptive method is based on an error analysis that gives the expected relative velocity error as a function of velocity and scale when using a certain derivative approximation to estimate the brightness gradient. The authors use this to detect those points whose velocity estimates can not be improved at a finer scale, i.e., points where the coarse scale is optimal. The motion vectors of the corresponding points at the finer scale are then simply set to the interpolated values from the coarser scale and do not participate in the iteration at this level. In the light of the discussion in the preceding sections, it appears straightforward to formulate a tensor field version of this method.

⁶See [99] for a discussion of the biological aspects of this and other coarse-to-fine strategies for motion computation.
A SIGNAL CERTAINTY APPROACH TO SPATIOTEMPORAL FILTERING AND MODELLING

The idea of accompanying a feature estimate with a measure of its reliability has been central to much of the work at the Computer Vision Laboratory. In recent years a formal theory for this has been developed, primarily by Knutsson and Westin, [72, 69, 68, 70, 103, 106, 105]. Applications range from interpolation (see the above references) via frequency estimation [71] to phase-based stereopsis and focus-of-attention control [101, 107]. In this chapter a review of the principles of normalized convolution (NC) is presented spiced with some new results. This is followed by a description and experimental test of a computationally efficient implementation of NC for spatiotemporal filtering and modelling using quadrature filters and local structure tensors. The experiments show that NC as implemented here gives a significant reduction of distortion caused by incomplete data. The effect is particularly evident when data is noisy.

3.1 Background

Normalized convolution (NC) has its origin in a 1986 patent [67] describing a method to enhance the degree of discrimination of filtering by making operators insensitive to irrelevant signal variation. Let $b$ denote a vector-valued filter and $f$ a corresponding vector-valued signal (e.g., a representation of local orientation). Suppose that the filter is designed to detect a certain pattern of vector direction variation irrespective of signal magnitude variation. A simple way of eliminating interference from magnitude variations would be to normalize the signal vectors. This is unfortunately not a very good idea. The magnitude of the signal contains information about how well the local neighbourhood is described by the signal vector model. This information is lost in the normalization. A special case is when the signal magnitude is zero – then a default model has to be imposed on the local neighbourhood. The patented procedure that
provides a solution to the problem is outlined in [43]:

“The method is based on a combination of a set of convolutions. The following four filter results are needed:

\[ s_1 = \langle b, f \rangle \]  \hspace{1cm} (3.1)
\[ s_2 = \langle b, ||f|| \rangle \]  \hspace{1cm} (3.2)
\[ s_3 = \langle ||b||, f \rangle \]  \hspace{1cm} (3.3)
\[ s_4 = \langle ||b||, ||f|| \rangle \]  \hspace{1cm} (3.4)

where \([\cdot, \cdot]\) denotes the magnitude of the filter or the data. The output at each position is written as an inner product \(\langle \cdot, \cdot \rangle\) between the filter and the signal centered around the position. [Formally this actually corresponds to a correlation between signal and filter but the use of the term convolution has stuck – the difference between the terms is just that in the convolution case the filter is reflected in its origin before the inner product is taken.]

The first term, \(s_1\), corresponds to standard convolution between the filter and the data. The fourth term, \(s_4\), can be regarded as the local signal energy under the filter. As to the second and third term, the interpretation is somewhat harder. The filter is weighted locally with corresponding data producing a weighted average operator, where the weights are given by the data giving a “data dependent mean operator”. For the third term it is vice versa. The mean data is calculated using the operator certainty as weights producing “operator dependent mean data”.

The four filter results are combined into

\[ s = \frac{s_4s_1 - s_2s_3}{s_4^2} \]  \hspace{1cm} (3.5)

where \(\gamma\) is a constant controlling the model selectivity. This value is typically set to one, \(\gamma = 1\). The numerator in Equation (3.5) can consequently be interpreted as a standard convolution weighted with the local “energy” minus the “mean” operator acting on the “mean” data. The denominator is an energy normalization controlling the model versus energy dependence of the algorithm.”

The procedure is referred to as a consistency operation, since the result is that the operators are made sensitive only to signals consistent with an imposed model. It was not until quite recently [106] that it was realised that the above method actually is a special case of a general method to deal with uncertain or incomplete data, normalized convolution (NC).
3.2 Normalized and differential convolution

Suppose that we have a whole filter set \( \{ b_k \} \) to operate with upon our signal \( f \). It is possible to regard the filters as constituting a basis in a linear space \( B = \text{span}\{ b_k \} \), and the signal may locally be expanded in this basis. In general, the signal cannot be reconstructed from this expansion since it typically belongs to a space of much higher dimension than \( B \). A Fourier expansion is an example of a completely recoverable expansion, which is further simplified by the basis functions (filters) being orthogonal.

There are infinitely many ways of choosing the coefficients in the expansion when the filters span only a subspace of the signal space. A natural and mathematically tractable choice, however, is to minimise the orthogonal distance between the signal and its projection on \( B \). This is equivalent to the linear least–squares (LLS) method.

Let us formulate the LLS method mathematically. Choose a basis for the signal space and expand the filters and the signal in this basis. Assume that the dimension of the signal space is \( N \), and that we have \( M \) filters at our disposal. The coordinates of the filter set may be represented by an \( N \times M \) matrix \( B \) and those of a scalar signal \( f \) by an \( N \times 1 \) matrix \( F \). The expansion coefficients may be written as an \( M \times 1 \) matrix \( \tilde{F} \). All in all we have

\[
B = \begin{bmatrix} b_1 & b_2 & \ldots & b_M \end{bmatrix} \quad F = \begin{bmatrix} f_1 \\
\vdots \\
f_N \end{bmatrix} \quad \tilde{F} = \begin{bmatrix} \tilde{f}_1 \\
\vdots \\
\tilde{f}_M \end{bmatrix}
\]

We assume that \( N \geq M \).

The LLS problem then consists in minimising

\[
E = \| BF - F \|^2
\]  

(3.6)

One often has a notion of reliability of measurement – as was indicated above the direction of vector-valued signals may carry a representation of a detected feature, whereas the magnitude indicates the reliability of that statement. When a measurement is unreliable, there is no point in minimising the projection distance for the corresponding element. On the other hand, one wants small distortion for the reliable components. This leads to the weighted linear least–squares (WLLS) method, with an objective function

\[
E_W = \| W(BF - F) \|^2
\]  

(3.7)

where \( W = \text{diag}(w) = \text{diag}(w_1, \ldots, w_N) \) is an \( N \times N \) diagonal matrix with the reliability weights. Letting \( A^* \) denote complex conjugation and transposition of a matrix \( A \) one

\[
E_W = \| W(BF - F) \|^2
\]  

(3.7)

where \( W = \text{diag}(w) = \text{diag}(w_1, \ldots, w_N) \) is an \( N \times N \) diagonal matrix with the reliability weights. Letting \( A^* \) denote complex conjugation and transposition of a matrix \( A \) one
finds
\[ E_W = (\tilde{F}'B' - F') W^2 W(B\tilde{F} - F) = \cdots = \]
\[ = \tilde{F}'B' W^2 B\tilde{F} - 2\tilde{F}'B' W^2 F - F'W^2 F = \tilde{F}'G\tilde{F} - 2\tilde{F}'x - c \]

Since \( G \) is positive definite we may use a theorem that states that any \( \tilde{F} \) that minimises \( E_W \) also satisfies the linear equation

\[ \tilde{F}'G\tilde{F} = x \]

Consequently

\[ \tilde{F} = G^{-1}x = (B'W^2B)^{-1}B'W^2F \] (3.8)

Introducing the notation \( \mathbf{ab} = \text{diag}(\mathbf{a}) \mathbf{b} \) for element-wise vector multiplication we may rewrite this in the language of convolutions

\[
\tilde{F} = \begin{bmatrix}
\langle w^2 \mathbf{b}_1, \mathbf{b}_1 \rangle & \cdots & \langle w^2 \mathbf{b}_1, \mathbf{b}_M \rangle \\
\vdots & \ddots & \vdots \\
\langle w^2 \mathbf{b}_M, \mathbf{b}_1 \rangle & \cdots & \langle w^2 \mathbf{b}_M, \mathbf{b}_M \rangle
\end{bmatrix}^{-1} \begin{bmatrix}
\langle w^2 \mathbf{b}_1, \mathbf{f} \rangle \\
\vdots \\
\langle w^2 \mathbf{b}_M, \mathbf{f} \rangle
\end{bmatrix} 
\] (3.9)

It turns out to be profitable to decompose the weight matrix into a product \( W^2 = \mathbf{AC} \) of a matrix \( \mathbf{C} = \text{diag}(\mathbf{c}) \) containing the data reliability and a second diagonal matrix \( \mathbf{A} = \text{diag}(\mathbf{a}) \) with another set of weights for the filter coefficients, corresponding to a classical windowing function. Consequently a realised filter is regarded as a product \( \mathbf{ab} \) of the windowing function \( \mathbf{a} \) and the basis function \( \mathbf{b} \). In this perspective it is clear that Equation (3.9) is unsatisfactory, but fortunately a little algebra does the job:

\[ \langle w^2 \mathbf{b}_i, \mathbf{b}_j \rangle = \langle \mathbf{ac} \mathbf{b}_i, \mathbf{b}_j \rangle = \langle \mathbf{ab}_i, \mathbf{cb}_j \rangle = \langle \mathbf{ab}_i, \mathbf{c} \rangle \]

and

\[ \langle w^2 \mathbf{b}_i, \mathbf{f} \rangle = \langle \mathbf{ac} \mathbf{b}_i, \mathbf{f} \rangle = \langle \mathbf{ab}_i, \mathbf{cf} \rangle \]

We arrive at a final expression for normalized convolution

\[
\tilde{F} = \begin{bmatrix}
\langle \mathbf{ab}_1 \mathbf{b}_1^*, \mathbf{c} \rangle & \cdots & \langle \mathbf{ab}_1 \mathbf{b}_M^*, \mathbf{c} \rangle \\
\vdots & \ddots & \vdots \\
\langle \mathbf{ab}_M \mathbf{b}_1^*, \mathbf{c} \rangle & \cdots & \langle \mathbf{ab}_M \mathbf{b}_M^*, \mathbf{c} \rangle
\end{bmatrix}^{-1} \begin{bmatrix}
\langle \mathbf{ab}_1, \mathbf{cf} \rangle \\
\vdots \\
\langle \mathbf{ab}_M, \mathbf{cf} \rangle
\end{bmatrix} 
\] (3.10)

Note that we must design \( M(M+1)/2 \) new outer product filters \( \mathbf{ab}_i \mathbf{b}_j^* \) in addition to the original \( M \) filters \( \mathbf{ab}_i \). The same expression as the above may be derived from tensor algebra regarding the process as a change of coordinate system in a linear space with a
metric whose coordinates are $\mathbf{a}$ in the original basis. One then finds that $\mathbf{G}$ contains the coordinates of the metric expressed in the basis dual to $\{\mathbf{b}_i\}$, whereas $\mathbf{G}^{-1}$ is the metric coordinates expressed in the original basis. The dual basis consists of the operators $\{\mathbf{b}^j\}$ for which $\langle \mathbf{b}^i, \mathbf{b}_j \rangle = \delta_j^i$. From the tensor algebra perspective Equation (3.10) describes a coordinate transformation of the signal from $\{\mathbf{b}^j\}$ to $\{\mathbf{b}_i\}$. To be able to compare the output of the normalized convolution with a standard convolution it is necessary to transform back to the dual basis. This is achieved by operating with the metric on $\tilde{\mathbf{F}}$, but since the varying reliability of data has now been compensated for, the transformation is achieved by using a matrix

$$
\mathbf{G}_0 = \begin{bmatrix}
\langle \mathbf{a} \mathbf{b}_1, \mathbf{b}_1^1 \rangle & \cdots & \langle \mathbf{a} \mathbf{b}_1, \mathbf{b}_M^1 \rangle \\
\vdots & & \vdots \\
\langle \mathbf{a} \mathbf{b}_M, \mathbf{b}_1^1 \rangle & \cdots & \langle \mathbf{a} \mathbf{b}_M, \mathbf{b}_M^1 \rangle 
\end{bmatrix} \quad (3.11)
$$

where $\mathbf{1} = (1, 1, \ldots, 1)^T$. This matrix may be precomputed since it is independent of the data weights.

Of course, the resultant output from the NC computation must be accompanied by a corresponding certainty function. There are three factors that influence this function: the input certainty, the numerical stability of the NC algorithm, and independent certainty estimates pertaining to new feature representations constructed from the NC filter responses$^1$. The numerical stability of the algorithm has to do with the nearness to singularity$^2$ of the metric $\mathbf{G}$. This is quantified by the condition number$^{[42]}$

$$
\kappa(\mathbf{G}) = \|\mathbf{G}\| / \|\mathbf{G}^{-1}\| 
$$

It is easily verified that $1 \leq \kappa(\mathbf{G}) \leq \infty$ for $p$-norms and the Frobenius norm. From this, we arrive at a possible certainty measure that takes into account the magnitude of the input certainty function

$$
c(\mathbf{G}) = \frac{1}{\|\mathbf{G}_0\| / \|\mathbf{G}^{-1}\|} \quad (3.12)
$$

The total output certainty function is then a product of $c(\mathbf{G})$ and the feature representation certainty.

To summarise, what have we achieved by this effort? From the original signal we have at each of the original positions arrived at an expansion of the signal in the basis $\{\mathbf{b}_i\}$ within a window described by $\mathbf{a}$, the applicability function, given a data weighting function $\mathbf{c}$, the certainty function. In this way we have imposed a model onto the original

$^1$For instance, an estimate of dominant orientation may be accompanied by a certainty function $c = \frac{\lambda_2}{\lambda_1}$.

$^2$We will here not discuss the possible use of generalised inverses, such as the Moore–Penrose pseudo-inverse, but see, e.g., [79, 82].
signal and by transforming back to the dual basis using \( G_0 \), the result is as is we had applied standard filtering on this model.

What is the relation between normalized convolution and the consistency operation described earlier? Suppose that the linear space \( \mathcal{B} \) spanned by \( \{ b_k \} \) is divided into two subspaces \( S \) and \( L \), that are orthogonal in the standard Euclidean metric. This orthogonality relation may no longer apply under the data reliability dependent metric \( G \). The result is that, to filter a signal using the NC approach, one may have to expand the signal in more basis functions than those which are of primary interest, the reason being that the new metric induces correlations between previously orthogonal basis functions. If not enough basis functions are used, the signal model may become inaccurate and the resultant filtering output misleading. Suppose that we are interested in filtering the signal with the basis functions in \( S \), but want to include the ones in \( L \) to model the signal. It is then possible to reduce the size of the metric matrix to be inverted in normalized convolution from the original \( M \times M \) to \( \text{Dim}(S) \times \text{Dim}(S) \). This is described in detail in [103]. Just as \( B \), the basis coordinate matrix may be decomposed, \( B = (S L) \). When \( \text{Dim}(L) = 1 \) we obtain the following expression for the projection of \( f \) on \( S \)

\[
\tilde{f}_S = (L' ACLS' ACS - S' ACCL' ACS)^{-1} (L' ACLS' A - S' ACCL' A) CF \quad (3.13)
\]

For the important case \( L = (1, 1, \ldots, 1)^T \) (DC component), this may in terms of convolutions be written

\[
\tilde{f}_S = G_S^{-1} \begin{bmatrix}
\langle a, c \rangle \langle ab_1, cf \rangle - \langle ab_1, c \rangle \langle a, cf \rangle \\
\vdots \\
\langle a, c \rangle \langle ab_{\text{Dim}(S)}, cf \rangle - \langle ab_{\text{Dim}(S)}, c \rangle \langle a, cf \rangle 
\end{bmatrix} \quad (3.14)
\]

with

\[
G_S(i, j) = \langle a, c \rangle \langle ab_i, b^*_j, c \rangle - \langle ab_i, c \rangle \langle ab_j, c \rangle \quad (3.14')
\]

This is referred to as normalized differential convolution (NDC). If we have a single filter \( ab \), insensitive to constant signals, and with the property that \( bb^* = (1, 1, \ldots, 1)^T \), we obtain

\[
f_S = \frac{\langle a, c \rangle \langle ab, cf \rangle - \langle ab, c \rangle \langle a, cf \rangle}{\langle a, c \rangle^2 - |\langle ab, c \rangle|^2} \quad (3.15)
\]

This is almost Equation (3.5) with \( \gamma = 2 \).

### 3.3 NDC for spatiotemporal filtering and modelling

In this section we will apply NDC in its most simple form, Equation (3.15), to the problem of estimating and modelling a local spatiotemporal neighbourhood. The model we
use is the local structure tensor reviewed in Section 1.4 and already extensively used in this thesis. Recall that it is estimated as a linear combination of six basis tensors with coefficients equal to the magnitude of six corresponding quadrature filter responses. The question now arises where and how to apply normalized convolution. One possibility is to use the six quadrature filters plus a low-pass filter (or more accurately, their corresponding basis functions, yet to be defined) as our signal model space. However, this is for efficiency reasons not possible, since it would require \( 7 \times 8/2 = 28 \) new complex outer-product filters and inversion of a \( 6 \times 6 \) complex matrix at each position. Instead we have to resort to a much more modest approach, namely to perform the NC at the quadrature filter level. The quadrature filter consists of an even real and an odd imaginary part that are reciprocal Hilbert transforms. With a supplementary constant basis function we consequently have at our disposal three basis functions. Being complex, the quadrature filter may be written
\[
h(x) = m(x) \exp[i\phi(x)] = m(x)(\cos\phi(x) + i\sin\phi(x)), \quad \text{where } m(x) \text{ is the magnitude and } \phi(x) \text{ the phase.}
\]
It is natural to let \( m \) become the applicability, and the real and imaginary parts of \( \exp[i\phi(x)] \) the two non-constant basis functions in NC. With three basis functions we have to generate four new outer-product filters \( m(x), m(x)\cos^2\phi(x), m(x)\sin^2\phi(x), \text{ and } m(x)\cos\phi(x)\sin\phi(x) \). Of course, the trigonometric identity \( \sin^2\phi + \cos^2\phi = 1 \) makes it possible to dispose of one of the filters, so we are left with three extra. Since signal and certainty data must be filtered separately, the grand total is \( 5/2 \times 2 = 5 \) times as many 3D-convolutions in NC as in the original algorithm. An equivalent formulation uses two complex basis functions \( \exp[\pm i\phi(x)] \) plus a constant function which also leads to three new (real) filters. Note that when a real-valued signal is expanded in such a basis, the optimal coefficients of the two complex basis functions will necessarily be equal except for a complex conjugation, so the number of free parameters are the same as in the case of real basis functions.

It is actually possible to reduce the number of convolutions further by modelling the signal with a single complex basis function \( b = \exp[i\phi(x)] \) plus a constant function. This was suggested by Westelius [101] based on experimental results, but no theoretical motivation was given. Since it may not be apparent why it should be useful to model a real signal with a single complex basis function, a simple hand-waving argument will be provided.

With two complex basis functions \( b \) and \( b^* \) we get a metric (cf. Equation (3.14))
\[
G_S = \left[ \begin{array}{ccc}
\langle a, c \rangle^2 - |\langle ab, c \rangle|^2 & \langle a, c \rangle \langle ab^2, c \rangle - \langle ab, c \rangle^2 & \langle a, c \rangle^2 - |\langle ab, c \rangle|^2 \\
\text{conjug.} & \langle a, c \rangle^2 - |\langle ab, c \rangle|^2 & \langle a, c \rangle^2 - |\langle ab, c \rangle|^2 \\
\end{array} \right]
\]
The off-diagonal filter \( ab^2 \) has a Fourier transform that essentially is a shifted version of the transform of the quadrature filter, its centre frequency being approximately twice that of the quadrature filter. Consequently it will be quite insensitive to low-frequency components of the certainty function. The three filters \( a, ab \) and \( ab^2 \) actually cover the Fourier space symmetrically—the first one being a lowpass filter, the second a band-pass...
pass filter and the third a highpass filter. Now, the certainty function has one particular characteristic: it is positive, which means that its DC-component is always the largest Fourier coefficient. As a result, the diagonal term $(a, c)^2$ tends to be larger than the others. In fact, a considerable variance in the certainty function combined with a low mean is required for the off-diagonal elements to become significant. In our present work the certainty function is a binary valued (0/1) spatial mask with zero certainty in regions that we want to disregard. We want to be able to filter close to the certainty discontinuity and to interpolate results across narrow zero-certainty gaps. Figure 3.1 shows what happens at a certainty edge. We used a realisation of the lognormal filter of Figure 1.4. The conclusion is that the non-diagonal elements are insignificant for positions outside the zero-certainty region\(^3\). Keeping only the diagonal elements, it is then easily seen that the NDC formula reduces to Equation (3.15).

\[ \begin{array}{c}
\text{Figure 3.1: Left: Certainty function. Right, solid line: } (a, c) \\
\text{Right, dashed: } (ab, c) \\
\text{Right, dotted: } (a b^2, c) \\
\end{array} \]

Having thus introduced the use of a single complex basis function, it is straightforward to apply the NDC formula Equation (3.15), once we have generated the applicability filter $a = m(x)$. For efficiency reasons this should be implemented as a separable filter. Since the applicability function is closely related to the quadrature filter we look for an implementation that can take advantage of this relationship. This is important for two reasons. Firstly, the design of the separable filters is done by optimisation (cf. Section 1.4.2) so generally there is a certain discrepancy between the resulting filter and the ideal 3D function. Two completely independent optimisations, one for the applicability, one for the quadrature filter, do not take into account the fact that the applicability func-

\[^3\text{An equivalent statement is that the basis functions } 1, \exp[i\phi(x)] \text{ and } \exp[-i\phi(x)] \text{ remain practically orthogonal in the new metric } ac.\]
tion should be equal to the magnitude of the realised quadrature filter. Secondly, there is a possibility of further reducing the number of operations. An important observation (cf. Section 1.4.2) is that the 3D quadrature filter may be decomposed into two orthogonal one-dimensional lowpass filters and a (predominantly) one-dimensional quadrature filter. It would be nice to be able to use the two lowpass filters in the applicability filtering, just adding a third lowpass filter next to the 1D quadrature filter. However, there is one problem—the coefficients of the applicability filter must be positive, which adds an extra constraint to the optimisation of the lowpass filters. Although a detailed description of the optimisation process is outside the scope of this presentation, a short digression may be in place, [7]. The positivity constraint is implemented by adding a regularisation term to the original cost function. The original cost is a measure of the distance between the ideal filter function and the realised filter in the Fourier domain. The new term similarly represents the distance between a constant, slightly positive ideal function and the realised filter, but now in the spatial domain. Let the total cost function be

\[ E_{\text{tot}} = \alpha E_F + (1 - \alpha) E_{\text{spat}} \quad 0 \leq \alpha \leq 1 \]

The adaptive factor \( \alpha \) is decreased below one when the current realisation of the filter contains negative coefficients, and increased towards one whenever the coefficients are all positive. The spatial cost function includes an adaptive weighting function which is increased wherever a coefficient is negative and decreased towards zero at spatial positions where the filter coefficient is non-negative. The idea is that the spatial ideal function should only influence the optimisation when (through \( \alpha \)) and where (through the spatial weighting function) the current coefficients are negative. The combined effect of the (non-linear) adaptive parameters is to force the iterative optimisation process to find the realisable filter with positive coefficients that is closest to the ideal filter function. The resultant 3D quadrature filters are actually not significantly inferior to the ones generated by the ‘unconstrained’ optimisation described in Section 1.4.2.

Figure 3.2 shows the pattern of sequential filtering that produces the quadrature and applicability filter responses, which may be compared with Figure 1.5. Note that we need two such structures—one to filter the signal-certainty product, and one to filter the certainty function itself. It is possible to generalise the scheme to more complex signal models that require additional filters, such as the \( \mathbf{a} \mathbf{b}^2 \)-filter that turns up when using two complex basis functions to model the signal. This filter is then of course also decomposed into the two lowpass filters followed by a third complex filter. To be able to use the NC result in the construction of the tensor we must transform back to the dual coordinates using the \( \mathbf{G}_0 \)-matrix of Equation (3.11). Since the quadrature filter is insensitive to constant signals this reduces to a multiplication by \( \langle a, 1 \rangle \), the sum of the applicability filter coefficients. To minimise computations the filters are normalized so that \( \langle a, 1 \rangle = 1 \).

From the NC quadrature filter responses and their output certainties, Equation (3.12), we
now proceed to construct the local structure tensor. The proper treatment of the certainty function related to the tensor construction procedure requires some modification of the presentation given in Section 1.4.1, Equations (1.4), (1.5) and (1.6), since a non-constant certainty function can be interpreted as a change of basis tensors $N_k$ and their duals $M_k$.

To take into account the filter response certainties $c_k$, we set

$$T_{est} = \sum_{k} q_k |M^f(\{c_k\}) = (\sum_{i} c_i N_i \otimes N_i)^{-1} \sum_{k} q_k c_k N_k$$

(3.16)

The numerical stability of the procedure depends on the nearness to singularity of the $6 \times 6$-matrix $[\sum c_i N_i \otimes N_i]$. We may consequently use its condition number to construct a certainty function for the tensor estimation, in exactly the same way as in Equation (3.12). The computational burden of inverting a $6 \times 6$-matrix at each spatial position is considerable. To avoid unnecessary operations the relative sizes of the filter certainties should be monitored so that precomputed matrices are used whenever the certainty variation is sufficiently isotropic. In this way the truly hard work is generally limited to a small fraction of the positions in each frame.
3.4 Experimental results

To determine the performance of the NDC implementation we used an onion-shaped 64-cube 3D volume, Figure 3.3 (left), for which the actual dominant 3D orientation is known at each position. We did not investigate the performance at the filter level, and, in particular, the phase behaviour was not accounted for. [Recall that only the magnitude of the filter response is used when constructing the tensor.] However, these properties have been thoroughly investigated by Westelius [101] for one-dimensional quadrature filters. The orientation estimation test was done by multiplying each 2D image with a certainty function set to zero at a number of rows in the central part and to one everywhere else. The idea is that we want to disregard the values in the zero-certainty slice when computing the 3D orientation outside the slice. Of course, by setting the signal value to zero we introduce an artifact that the standard convolution cannot handle. The question is how well the NDC algorithm will do. We chose to look at the 3D orientation half-way through the volume, at frame 32. The dominant 3D orientation at each spatial position was estimated from the tensor and compared with the ’actual’ value. An average angular error for each row was computed by averaging over the columns. The test was repeated for three different levels of white noise, $\mu$ dB, 10 dB, and 0 dB, Figures 3.3 – 3.15. As is seen, the NDC implementation quite successfully manages to capture the correct orientation at the mask border whereas the standard algorithm estimates an ’incorrect’ signal orientation several pixels outside the border due to the artifact edge generated by the zero-mask. Also, note that the imposed signal model appears to have a very beneficial effect on the performance for noisy signals.

3.5 Discussion

Despite several very successful applications, including the new ones in this thesis, we feel that the full potential of the signal/certainty philosophy is yet to be exploited. A consistent signal/certainty treatment should be an integrated part of the processing at all levels of a robot vision system, including high-level segmentation, recognition, and decision-making. Adaptive signal processing (‘learning’) in combination with normalized convolution (or a similar concept) has the potential to provide very efficient high-level components. Some quite promising work in this direction has been carried out [19].
Figure 3.3: Left: Slice 32 from the test sequence. Right: Orientation error in degrees for each row averaged over all columns.

Figure 3.4: Output certainty functions. Left: Mask width 4 pixels. Right: Mask width 20 pixels.
Figure 3.5: Noise-free sequence. Mask width 4 pixels. Average orientation error in degrees for each row. *Left:* Standard convolution. *Right:* Normalized convolution.

Figure 3.6: Noise-free sequence. Mask width 4 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. *Left:* Standard convolution. *Right:* Normalized convolution.
Figure 3.7: Noise-free sequence. Mask width 20 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. *Left*: Standard convolution. *Right*: Normalized convolution.

Figure 3.8: *Left*: Slice 32 from the test sequence corrupted with white noise at 10dB SNR. *Right*: Orientation error in degrees for each row averaged over all columns.
Figure 3.9: 10dB SNR. Mask width 4 pixels. Average orientation error in degrees for each row. Left: Standard convolution. Right: Normalized convolution.

Figure 3.10: 10dB SNR. Mask width 4 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. Left: Standard convolution. Right: Normalized convolution.
Figure 3.11: 10dB SNR. Mask width 20 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. Left: Standard convolution. Right: Normalized convolution.

Figure 3.12: Left: Slice 32 from the test sequence corrupted with white noise at 0dB SNR. Right: Orientation error in degrees for each row averaged over all columns.
Figure 3.13: 0dB SNR. Mask width 4 pixels. Average orientation error in degrees for each row. \textit{Left}: Standard convolution. \textit{Right}: Normalized convolution.

Figure 3.14: 0dB SNR. Mask width 4 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. \textit{Left}: Standard convolution. \textit{Right}: Normalized convolution.
Figure 3.15: 0dB SNR. Mask width 20 pixels. Errors inside mask deleted. Average orientation error in degrees for each row. Left: Standard convolution. Right: Normalized convolution.
MOTION SEGMENTATION AND TRACKING

In this chapter we present a simple application within the realm of active vision, using concepts developed in the previous chapters. Active vision\(^1\) [4, 2, 8, 9, 10, 11, 3, 27, 30] is a paradigm that regards vision as a process with a purpose, a closed loop involving the observer and the environment with which it interacts. Control of sensing is stressed—the current goal of the system dictates attention. Active vision also emphasizes the importance of proprioception in facilitating and resolving ambiguities in the perception of the environment. This involves knowledge of the observer’s own position, motion, as well as accurate models of sensors and actuators.

We present computationally efficient algorithms for smooth pursuit tracking of a single moving object. We use knowledge of pan and tilt joint velocities to compensate for camera-induced motion when tracking with a pan-tilt rotating camera. In a second algorithm we allow camera translation and assume the background motion field in the vicinity of the target can be fitted to a single parameterised motion model.

Most computer vision tracking algorithms derive the motion information necessary for the pursuit from observation of target position only, which unfortunately is very difficult to obtain with precision. The reason that observations of velocity are not used is of course the computational cost of accurate measurement—most real-time algorithms match pairs of consecutive images and then it is just not possible to obtain accurate and stable velocity estimates. Advances in signal processing theory with designs of highly efficient sequential and recursive filtering schemes for velocity estimation challenge the old established ‘quick—and—dirty’ approaches to motion tracking. In our work we use the multi-resolution local structure tensor field method described in Chapter 2 to obtain high-quality estimates of velocity which we then use actively in prediction and segmentation. The algorithms readily incorporate the signal certainty/modelling formalism developed in the previous chapter.

4.1 Space–variant sensing

A visual system, whether biological or artificial, that has the ambition to provide its host with useful information in a general setting, must be able to process information at multiple spatial and temporal resolutions. Discrimination of detailed shape and texture requires very high spatial sensor resolution, whereas reliable estimation of high speeds demands large receptive fields. To cover the broad spectrum of spatial and temporal frequencies that appear in the visual input that is relevant to higher animals or autonomous robots a whole range of resolution levels is needed. The problem is that any information processing system has a limited capacity, which means that an implementation must make some sort of compromise. The eyes of higher animals have developed into space-variant *foveal* sensors which have decreasing spatial resolution towards the periphery. The density of sensory elements is several orders of magnitude higher in the centre of the field of view, the *fovea*, than in the periphery. Correspondingly, in the primary visual cortex there are several times as many neurons engaged in processing foveal signals as there are neurons dedicated to peripheral information. To compensate for the lack of spatial resolution in the periphery, the visual system constantly shifts its gaze direction to bring peripheral stimuli of potential interest into the centre of the field of view for scrutiny. This is done by means of *saccades*, fast ballistic eye movements to foveate the target. The concept of a space-variant sensor, or fovea for short, has been carried over to the field of computer vision. There exist several hardware approaches, e.g., [87, 84]. A software approach is to compute a complete multi-resolution pyramid, but to throw away most of the data. An example is the log–Cartesian fovea [29], where a multi-resolution region-of-interest (ROI) is defined within the pyramid. From each level an equal size window is kept, which means that the finer levels cover a smaller part of the field of view than the coarser levels. There is no restriction to the position of the centre of the ROI, and consequently no explicit gaze direction change is needed for high resolution processing in the periphery. Of course, image processing on this type of data structure can be quite complicated. We use a simple version [101] of the log–Cartesian fovea, where the ROI is always at the centre of the field of view and of constant size, Figure 4.1.

4.2 Control aspects

We now give a short account of the basics of human voluntary smooth pursuit tracking [51, 28, 24, 83, 74, 108]. The tracking process always involves an initial shift of attention to the stimulus, or target. If the target is not initially foveated, i.e., positioned in the centre of the field of view, a saccade is produced. The eyes then accelerate to their final speed within approximately 120 msec. Interestingly, this final speed is ap-
proximately 10% less than the speed of the target, so that the eyes tend to lag behind the target. This is compensated for by occasional saccades that recenter the target in the fovea. The principal stimulus for pursuit movements is retinal slip, i.e., the translational image velocity of the target, though there is also evidence for a weak position error component for small sudden target offsets from the fovea, with the pursuit velocity being proportional to the offset. A conventional negative feedback controller, Figure 4.2, is not able to accurately model the characteristics of the human smooth pursuit system. The reason is that the model is unstable with the high gain and long delays that are found experimentally. High gain, of course, is necessary to obtain good accuracy in pursuit, but can be very problematic in systems with long delays. To overcome this problem, evolution has developed a quite different concept, Figure 4.3, where an internal (adaptive) model of the ocular motor system is used to predict the eye velocity and subsequently to reconstruct the target velocity. This is a manifestation of the fact that 'the brain knows the body it resides in' [24].

Next, we consider the design of the controller for a machine vision smooth pursuit tracker. Although control issues are not central to our work, there are at least two aspects of a motion estimation based smooth pursuit tracker that distinguish it from most other trackers, so a certain digression may be justified. Most algorithms, whether for passive or active tracking, only use observations of position. Furthermore, the delay of the observations is regarded as negligible. When these conditions apply there are standard algorithms for state reconstruction, e.g., the so-called α–β– and α–β–γ–filters, [12]. We want to use both position and velocity observations, and cannot ignore that the estimates are delayed several sampling interval units. We define the overall goal of our control algorithm as to keep the centroid of the target motion field stationary in the centre of the field of view. Note that this consists of two conflicting sub-goals, namely to stabilize
Figure 4.2: A (too) simple model of the human smooth pursuit system. $P(s)$ is the transfer function of the ocular motor system, $X$ and $E$ the target and eye positions, respectively. $K \approx 0.9$, $T \approx 0.04$ sec. With the experimentally verified delay $\tau \approx 0.13$ sec, this system is unstable.

Figure 4.3: A more accurate model of the human smooth pursuit system (disregarding interaction with the saccade system). This appears to first have been proposed by Young, [109]. The system utilizes an internal (adaptive) model of the ocular motor system to predict the eye velocity and uses this to reconstruct the target velocity. This effectively turns the system into an unconditionally stable feed–forward controller. After [83].
disproportionately slow for large movements. Secondly, even with fast motors, there is always a recovery period after each saccade during which no observations can be made, due to the fact that the temporal buffers of the spatiotemporal filters must be refilled. By including a weak smooth position error compensation we can avoid some saccades and the associated information loss, without compromising image stability too much. Since there is no a priori reason for a coupling between motion in horizontal and vertical directions, we use separate algorithms for each spatial dimension. Let \( x(kT) \) denote the position of the target at time \( t = kT \), where \( T \) is the sampling interval. Assuming a slow acceleration, we have approximately

\[
\begin{align*}
    x((k+1)T) &= x(kT) + T \dot{x}(kT) \\
    \dot{x}((k+1)T) &= \dot{x}(kT) + T \ddot{x}(kT)
\end{align*}
\]

A state model that assumes constant velocity is unrealistic. When one has to take into account planned target velocity changes, it is common to model acceleration as a stochastic drift process, e.g., an IAR process, [21]. The control is achieved by specifying camera head joint velocities, and we assume that the motor controller is significantly faster than the pursuit loop. With state variables \( X_1(k) = x(kT) \), \( X_2(k) = \dot{x}(kT) \), and \( X_3(k) = \ddot{x}(kT) \), we then arrive at the following closed-loop state equation

\[
X(k+1) = AX(k) + Bu(k) + v = \\
\begin{bmatrix}
1 & T & 0 \\
0 & 1 & T \\
0 & 0 & 1
\end{bmatrix} X(k) + \\
\begin{bmatrix}
0 \\
0 \\
v(k)
\end{bmatrix}
\]

where \( v(k) \) is white noise. We use state variable feedback

\[
u(k) = -LX(k) = -\begin{bmatrix} u_p & u_d & 0 \end{bmatrix} X(k)
\]

with \( u_p \) and \( u_d \) constants that determine the influence of positional and velocity errors, respectively. The state is reconstructed from observations by Kalman filtering. Now, because of the temporal buffer depth, observations of position and velocity are delayed several sampling intervals so we have to use a multi-step predictor. The observations are

\[
y(k) = CX(k) + \varepsilon = \\
\begin{bmatrix} 1 & 0 & 0 \\
0 & 1 & 0 \end{bmatrix} X(k) + \\
\begin{bmatrix} e_1(k) \\
e_2(k) \end{bmatrix}
\]

where \( e_1(k) \) and \( e_2(k) \) are white noise processes. The optimal linear \( m \)-step predictor is then given by (e.g., [5])

\[
\hat{X}(k|k-m) = A^m \hat{X}(k-m|k-m) + \sum_{i=k-m+1}^{k} A^{k-i} Bu(i-1)
\]

where the reconstructor is given by

\[
\hat{X}(k|k) = [I - KC][A\hat{X}(k-1|k-1) + Bu(k-1)] + Ky(k)
\]
Here $K$ is the $3 \times 2$ Kalman gain matrix, the properties of which depend on the noise processes $v$ and $e$, whose covariance matrices are

$$
R_1 = E[v(k)v(l)^T] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_v^2 \end{bmatrix} \delta_{kl} \quad R_2 = E[e(k)e(l)^T] = \begin{bmatrix} \sigma_{pos}^2 & 0 \\ 0 & \sigma_{vel}^2 \end{bmatrix} \delta_{kl}
$$

(4.6)

The (stationary) Kalman gain is then given by

$$
K = APC^T[CPC^T + R_2]^{-1}
$$

(4.7)

where the prediction error covariance matrix $P$ satisfies

$$
P = R_1 + APA^T - APC^T[CPC^T + R_2]^{-1}CPA^T \quad \text{[discrete matrix Riccati equation]}
$$

(4.8)

It is difficult to estimate the process variances $\sigma_v^2$, $\sigma_{pos}^2$ and $\sigma_{vel}^2$, so we regard them, as well as the feedback parameters $u_p$ and $u_d$, as free parameters that we have at our disposal to make the system behave appropriately.

### 4.3 Motion models and segmentation

#### 4.3.1 Segmentation

The process of motion-based image segmentation in a system with focus of attention control consists of two separate stages—an 'early' preattentive detection stage and a 'late' object recognition oriented attentive stage. The parallel preattentive process detects local structure in the motion field and thereby indicates where potentially interesting stimuli appear in the field of view. The local structures of interest are optical flow patterns characteristic to objects in motion, such as those shown in Figure 4.4. Indeed, neurons that are sensitive to this type of stimuli have been found in the primate visual system, [88, 31]. The problem of detecting independently moving objects with a moving observer is nontrivial, see, e.g., [26, 90]. Recently Fermüller and Aloimonos [2, 37, 34, 35], based on earlier work by Nelson [80], have found a class of motion constraints that, given bounds on egomotion, allows a moving observer to detect local motion patterns that can not originate from the motion of the observer. In a computer vision system the preattentive process may be implemented as a set of convolutions of the velocity estimates with symmetry operators [15, 104]. The magnitude of the filter responses defines a 'potential field' with local extrema at interesting points. It is possible to devise algorithms where the attention is attracted to successive interesting points,
4.3 MOTION MODELS AND SEGMENTATION

Figure 4.4: Characteristic motion vector patterns generated by solid objects in motion.

to a certain extent analogously to the interaction between a particle and a field of force [101].

Once a moving object has become the focus of attention, segmentation enters the second, attentive stage. The overall goal of this stage is usually regarded as to provide information for the inference of object shape and 3D depth and motion, a problem referred to as structure from motion. Considering the importance of this, it is not surprising that the amount of literature is absolutely enormous, see, e.g., [48, 46] and references therein. Most structure recovery methods rely on accurate estimation of the motion field. Aloimonos and Fermüller (above references) suggest robust methods based on qualitative properties of motion and bounds on egomotion.

Here we confine ourselves to a much more restricted task—to extract the 2D projection of a single moving rigid object, fit its 2D motion vectors to a parameterised model, and track it by smooth pursuit. The basic assumption we make is that, at least occasionally, the motion field in the immediate vicinity of the target can be fitted to a single parameterised model, Figure 4.5. When this applies, we can use a very simple procedure to

Figure 4.5: Basic assumption of the approach is that, at least occasionally, we can find a region in which the target and background motion fields each fit a single parameterised model.

extract the target. The idea is to estimate the background motion in an annular region surrounding the target, and then use this to predict motion in the region inside the annu-
The local motion estimates in the central region that are not well predicted by the annular motion model are interpreted as due to the target motion. We can formulate this as an iterative algorithm.

1. We assume that an estimate of target size and position is available. This defines a predicted target region. Initially this must be provided either by a preattentive motion detector or by top–down prediction.

2. Fit a motion model to a ring around the target region, and if the model has a small residual, use it to predict the motion in the target region. For each position, we compare the predicted value with the actual estimate, and if the error is small, we set a flag to mark that the estimate is consistent with the ring model.

3. If the prediction is satisfactory for most positions, we merge the ring and the central region to a new target region and repeat the process. The interpretation is that both the ring and the central region are completely contained within the background or the target.

4. If the ring model does not cover all data in the target region, we fit a motion model to the remaining estimates in the region. If this model gives a small residual we assume that we have found the target, otherwise we merge the ring and the central region to a new target region and repeat the process.

Figure 4.6 provides an illustration of two typical segmentation situations. The result of a successful segmentation process is a set of target motion parameters and the set of points whose motion estimates were used to estimate the parameters. We use the lowpass–filtered confidence values of the motion estimates in these points to construct a simple moving average model of the target. On those occasions when the segmentation process fails, i.e., when the region grows beyond a predetermined maximum size, we use the target model and predicted values of the motion parameters to pick out estimates.
in the original target region for model fitting. The use of prediction of velocity and position increases the robustness against interference from occlusion. Comparison between predicted and estimated motion parameters permits detection of incorrect parameter estimates which would correspond to unreasonable acceleration.

In the case of a pan/tilt rotating camera it is possible to use an alternative procedure based on elimination of the camera–induced background motion field. From Figure 4.7 it is easy to convince oneself that an accurate expression for the camera–induced displacement is given by

\[
\begin{pmatrix}
\Delta x \\
\Delta y \\
0
\end{pmatrix} = \omega \times r + \frac{1}{f} \hat{z} \cdot (\omega \times r) r
\]

where \( \omega = (\omega_x, \omega_y, 0)^T \) is the angular velocity vector. At each position we may construct a unit length spatiotemporal displacement vector \( \hat{u} \) and multiply each tensor \( \hat{T} \) with a factor that is a measure of the inconsistency between the tensor and the background displacement direction. One possible choice is a ‘soft threshold’

\[
f(\hat{u}, \hat{T}) = 1 - \exp\left( -\frac{\hat{u}^T \hat{T} \hat{u}}{||\hat{T}|| \sigma^2} \right)
\]

This method to eliminate self–induced background motion is actually similar to how we pick out tensors that are consistent with model predictions (see above)—it may be regarded as a kind of focus–of–attention. The basic idea is illustrated in Figure 4.8, where a cone symbolises the (hard or soft) threshold angle. With the use of motion elimination the extraction of the target in the rotating camera case is achieved by simply increasing the target region until no further data consistent with the target motion is found.
4.3.2 Discussion

The segmentation methods presented above are related to several other approaches for motion tracking with an active camera. Nordlund and Uhlin [81, 94] fit two consecutive frames to a global 2D displacement model (translation or affine transformation) and detect the centroid of the target displacement field from the residual image. The benefit of this method is that it is computationally efficient. An obvious drawback is the loss of locality, which implies very strong assumptions about the global structure of the scene. Also, the quality of tracking, i.e., the ability to stabilize the target on the image, is compromised by the lack of precise motion estimates.

Murray and Basu [76] use a modification of image subtraction combined with elimination of camera–induced motion to track a moving object with a pan/tilt rotating camera—they do not address the case of a translating camera. Again, the lack of accurate motion estimates prevents good image stabilisation.

Tölö [92], see also [86], uses a differential–based optical flow algorithm [95] which makes it possible to extract the target motion field by clustering—only fronto–parallel
motion is treated. Tölg, assuming a pan/tilt rotating camera, also compensates for camera–induced motion, but bases this on motion vector subtraction, which assumes that the true motion vector can be accurately estimated. However, it is known that in general only the local normal velocity (component in direction of spatial gradient) is available with some precision.

The question of how to represent the target region shape has been addressed by many authors in the field of image sequence coding, but has attracted comparatively little attention in tracking. The principal use of shape information in tracking is to implement a spatial weight mask—a kind of focus–of–attention. Meyer and Bouthemy [73] use a region descriptor (convex hull) to detect occlusion from region area size changes. Though their motion–based region segmentation method, [20], is unsuitable for real–time applications, the idea of a more structural region representation is conceivable also for sparse motion fields using tools from computational geometry, [93, 96].

4.3.3 Motion models

We have experimented with three different motion models—pure 2D translation, translation and expansion/contraction, and affine transformations. Concerning parameter estimation, computational efficiency was stressed—costly iterative methods were regarded as unacceptable. The pros and cons of this are discussed at the end of this section.

The case of pure translation is particularly simple, as illustrated in Figure 4.9. If a set of local structure tensors\(^2\) emanate from a single pure 2D translation, the sum of the tensors, \(T_{\text{sum}}\), will be of rank 2 with the eigenvector corresponding to the smallest eigenvalue pointing in the direction of spatiotemporal displacement. If the tensors do not come from a single translation, \(T_{\text{sum}}\) will have a significant third eigenvalue \(\lambda_3\). Consequently we can use the quotient \(\lambda_3/\text{Tr}T_{\text{sum}}\) as a measure of the deviation from pure translation.

When there is a significant velocity component orthogonal to the image plane, we have to take into account the perspective transformation. Restricting ourselves to a single spatial dimension, the perspective transformation equation may be written

\[
x = -f \frac{X}{Z}
\]

(4.9)

where \(f\) is the camera constant, \(x\) denotes the image coordinate, \(X\) the corresponding world coordinate, and \(Z\) the orthogonal distance from the camera lens to the object.

\(^2\)We use the rank 2 tensors \(\mathbf{T}\) introduced in Section 2.2.
Differentiating Equation (4.9) with respect to time, we find

$$\dot{x} = f \frac{X}{Z} \dot{Z} - f \frac{X}{Z}$$

(4.10)

Assuming that the variation in depth of the visible part of the target along the line of sight is small compared with the viewing distance (weak perspective), we conclude that the apparent velocity of a projected point is a linear function of its distance from the image coordinate centre. In Figure 4.10 we see that this means that the spatiotemporal velocity vectors converge at a point, which we may call the spatiotemporal focus-of-expansion, $x_{ST FOE}$. To find this point, one may use some kind of clustering scheme or proceed as follows. The spatiotemporal displacement vector at $x = (x, y, t)^T$ is parallel to $x - x_{ST FOE}$. Ideally the local structure tensor $\tilde{T}(x)$ has no projection in this direction, so that

$$\tilde{T}(x) (x - x_{ST FOE}) = 0$$

We can find a least squares optimal value of $x_{ST FOE}$ in a spatial region $\mathcal{R}$ by minimising

$$\mathcal{E} = \sum_{x \in \mathcal{R}} (x - x_{ST FOE})^T \tilde{T}(x) (x - x_{ST FOE})$$

$$= x_{ST FOE}^T \sum_{x} \tilde{T}(x) x_{ST FOE} - 2 x_{ST FOE}^T \sum_{x} \tilde{T}(x) x + \sum_{x} x^T \tilde{T}(x) x$$

Since $\sum \tilde{T}(x)$ is positive (semi–) definite, it follows that the optimal $x_{ST FOE}$ is a solution
to the linear equation system

$$\sum \tilde{T}(x) x_{STFOE} = \sum \tilde{T}(x) x$$

So in addition to the sum of tensors that we use for the pure translation model estimation, we need the vector sum \( \sum \tilde{T}(x) x \). The translation component of the motion can be found from the average displacement direction \( \frac{1}{n} \sum_{i=1}^{n} \tilde{T}(x)(x - x_{STFOE}) \).

An alternative way to proceed when modelling the motion field as a fronto–parallel translation plus motion orthogonal to the image plane is to make the ansatz

$$\begin{pmatrix} u \\ v \end{pmatrix} = a \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$

We will not describe this in detail, since it is a simple special case of our final approach, namely to model the motion field as an affine transformation

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$  \hspace{1cm} (4.11)

This type of motion model has recently become a popular choice for the segmentation stage of video sequence coding algorithms, e.g., [58, 73, 100, 33]. One reason for this is that it can be shown that the general motion of a planar surface patch under orthographic projection can be expressed as an affine transformation.
The linear transformation represents rotation in the image plane, depth motion and shear, while the constant part describes translation in the image plane.

Rewriting Equation (4.11) as a linear transformation of the spatiotemporal displacement vector \( \tilde{u} = (\Delta x, \Delta y, \Delta t) = (u, v, 1)^T \), we obtain

\[
\tilde{u} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = A \tilde{x} = \begin{pmatrix} a_{11} & a_{12} & u_0 \\ a_{21} & a_{22} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

(4.12)

Again noting that the local structure tensor at each position has no projection in the direction of spatiotemporal displacement, a least squares sense optimal affine motion model for an image region \( R \) satisfies

\[
A^{opt} = \arg \min_A \sum_{x \in R} \tilde{u}^T (x, y) \tilde{T} \tilde{u} (x, y) = \arg \min_A \sum_{x \in R} x^T A^T \tilde{T} A x
\]

(4.13)

Setting the partial derivatives with respect to the parameters to zero results in the following symmetric linear equation system

\[
\sum_{(x,y) \in R} \begin{pmatrix} x^2 t_{11} & xy t_{11} & xt_{11} \\ y^2 t_{11} & yt_{11} & yt_{11} \\ t_{11} & y t_{11} & y t_{11} \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ u_0 \end{pmatrix} + \begin{pmatrix} x t_{13} \\ y t_{13} \\ t_{13} \end{pmatrix} = 0
\]

(4.14)

where \( t_{ij} \) refers to the \((i, j)\) component of \( \tilde{T} \). In Appendix A.2.1 we present results of an experimental investigation of the accuracy of the affine parameter estimates\(^3\). We use \( \alpha-\beta \)-filters \([12]\) for prediction of the four target motion parameters \( a_{11}, a_{12}, a_{21}, \) and \( a_{22} \) during tracking.

Possibly the only reason for using linear least squares methods for motion parameter estimation is that they are computationally efficient, which of course is critical in real-time tracking. A well-known problem with linear least squares methods is their sensitivity to outliers. There exist several techniques, referred to as robust estimation, \([52, 45]\), where outliers are rejected in an iterative process. This typically consists in maximising a log likelihood function where the data set is considered to be divided into a Gaussian signal process and a non-Gaussian outlier noise process. There are also various clustering techniques, e.g., \([85, 57]\), that can be used as robust alternatives to linear least

\(^3\)Experimental results for the simpler motion models are not included—they are similar to the results for the affine model when applicable.
squares. Without going into details, let us just mention that it is completely straightforward to apply these methods to motion parameter estimation using the spatiotemporal constraints presented in this section. Any of these techniques typically requires several times as much work as linear least squares.

4.4 On the use of normalized convolution in tracking

Presently, we see two main applications of normalized convolution in active tracking. Firstly, it is possible to shorten the recovery period following a saccade by several sampling intervals by giving the old pre–saccade frames that are left in the temporal buffer a certainty value equal to zero. In this way a rough estimate of the post–saccade target velocity can be produced very quickly. Secondly, it is possible to integrate normalized convolution with focus–of–attention control by using zero–certainty masks to suppress interference from known structures, such as a robot arm or already modelled objects. Details of this are given in [103, 101, 107].

4.5 Experimental results

An advanced graphics simulation environment, [102, 101], was used for development and evaluation of the tracking routines. The tracker was implemented on a simulated standard Puma 560 robot with a mounted (stereo–) camera head, Figure 4.11. The use of a simulated environment has a number of advantages—it allows for a wide range of experimental situations, it provides true velocity and position which can be used in algorithm performance evaluation, and it makes you independent of special purpose hardware. Simulation can certainly never replace performance in 'the real world' as the final criterion of failure or success—particularly when there is a real-time constraint—but it is extremely useful in early stages of development.

The sequence in Figure 4.13 shows smooth pursuit with the robot initially approaching the target, then moving parallel to it, and finally stopping in front of the approaching target. Note how the robot's maneuvers are reflected in displacement of the target from the centre of the field of view. In Figure 4.14 the weights (confidence values) of the local motion estimates that were used to extract the target are superimposed on the target which is tracked with a rotating camera.

The code has been ported to the Aalborg University Camera Head Demonstrator, Figure 4.12. Although only a few experiments were carried out with the installed code, we
consider them sufficient as a ‘proof of concept’, see Figures 4.15 and 4.16. The image processing was done at a moderate 2 Hz on a Unix workstation by importing the video signal to the above mentioned graphics simulation package. The simulation package code communicated with the onboard camera head controllers via sockets to issue commands and receive actual camera head joint velocities and positions. We anticipate a more careful implementation not involving the graphics package to achieve at least 10 Hz.
4.5 EXPERIMENTAL RESULTS

**Figure 4.11**: Robot simulation. Left eye, right eye, and overview. The test platform consists of a simulated Puma 560 robot with a mounted stereo camera head.

**Figure 4.12**: The Aalborg University Camera Head Demonstrator system.
Figure 4.13: Simulated sequence with a translating camera. A box is transported on a conveyor belt. The small cross marks the centroid of the target motion field, whereas the large cross simply marks the centre of the image.
Figure 4.14: Simulated sequence. The confidence values of the extracted motion estimates are superimposed on the target.
Figure 4.15: ‘To and fro’—a real–time smooth–pursuit tracking sequence recorded at the Laboratory of Image Analysis, Aalborg University.
Figure 4.16: 'Walking the plank'—a real-time smooth-pursuit tracking sequence recorded at the Laboratory of Image Analysis, Aalborg University.
The thesis has treated certain aspects of efficient spatiotemporal filtering and modelling.

A novel energy–based multiresolution algorithm for estimation and representation of local spatiotemporal structure by second order symmetric tensors has been presented. The algorithm utilizes coarse–to–fine data validation to produce accurate and reliable estimates over a wide range of image displacements. It is highly parallelisable thanks to an efficient separable sequential implementation of 3D quadrature filters. Results show that the algorithm is capable of producing very accurate estimates of image velocity.

An efficient spatiotemporal implementation of a certainty–based signal modelling method called normalised convolution has been described and experimentally investigated. By using a novel iterative filter optimisation method to produce positive filter coefficients, we show that normalised convolution can be efficiently integrated with the sequential quadrature filtering scheme. We also describe how certainty information is incorporated in the construction of the local structure tensor. Experimental tests show that normalised convolution for signal interpolation in spatiotemporal sequences is very effective, due to the high information redundancy in the signal.

As an application of the above results, we have presented a smooth pursuit motion tracking algorithm that uses observations of both target motion and position for camera head control and motion prediction. We introduce a novel target motion segmentation method which assumes that the motion fields of the target and its immediate vicinity, at least occasionally, each can be modelled by a single parameterised motion model. We use the acquired and updated information about target position and velocity to extract motion estimates on those occasions when the segmentation algorithm fails. We also show how to eliminate camera–induced background motion in the case of a pan/tilt rotating camera. We provide excerpts from both simulated and real tracking sequences that show that the algorithms work.
The thesis probably creates more question-marks than it straightens out. Certainly a
great deal more work can be put into the study of the multi-resolution tensor field es-
timation algorithm. There may be more optimal ways to combine the different levels,
and it may also be useful to consider some local interaction at each level. In particular, a
study of relaxation techniques would be interesting. Considering applications, promis-
ing work on region-based motion segmentation for image sequence coding has recently
been done [33].

The efficient implementation of normalised convolution is an important result. As men-
tioned earlier, we think that the full potential of the signal/certainty philosophy is yet to
be exploited. For instance, there has to be work done on integrating certainty processing,
feature extraction and object recognition.

A straightforward refinement of the iterative target segmentation algorithm would be
to make the region growing process shape adaptable, most simply by computing the
principal axes of the motion field distribution. Of course, there are dozens of pos-
sible real-time applications other than smooth pursuit tracking that can use accurate
motion estimates. To name but a few: egomotion estimation, obstacle avoidance, fix-
ation, and homing. Also, as mentioned earlier, there are better ways to estimate mo-
tion model parameters than the linear least squares methods that we have used in this
work. It would be interesting to study the performance of, e.g., mixture models, [57],
and to see whether they can be efficiently implemented within the local structure ten-
sor framework—it seems likely that robust iterative methods converge faster when the
local estimates are of high precision with occasional outliers, than if they have a large
variance as is the case for the more primitive optical flow methods.

* * *
In this appendix we provide some of the more technical details and results of experimental tests that for continuity reasons were omitted in the main text. The mathematical results are all quite elementary, but it was felt that they should be included for the sake of completeness.

A.1 Local motion and tensors

A.1.1 Velocity from the tensor

There are two fundamental cases.

1. Moving point, Figure A.1. The spatiotemporal line traced out by the moving point has a tangent vector $v_{ST} = (v_x, v_y, 1)^T$ which is parallel to the eigenvector $\hat{e}_3 = (e_{31}, e_{32}, e_{33})^T$ corresponding to the smallest eigenvalue of the tensor. Consequently the velocity is given by

$$v = (v_x, v_y)^T = \frac{1}{e_{33}} (e_{31}, e_{32})^T$$

(A.1)

2. Moving line, Figure A.2. The moving line creates a plane in the spatiotemporal volume. In this case the true velocity can not be determined, since it is impossible to detect any velocity component parallel to the line. However, the component $v_{norm}$ normal to the line is easily recovered by noticing that its spatiotemporal counterpart $v_{norm}^{ST}$ is orthogonal to the eigenvector $\hat{e}_1$ corresponding to the largest eigenvalue, as well as to a vector $l = (e_{12}, -e_{11}, 0)^T$ parallel to the line of intersection of the motion plane and $t = t_0$. But

$$e_1 \times l = (-e_{12}e_{13}, -e_{12}e_{13}, e_{11}^2 + e_{12}^2)^T$$
so that
\[ v_{\text{norm}} = \frac{-e_{13}}{e_{11}^2 + e_{12}^2} (e_{11}, e_{12})^T \] (A.2)

since the temporal component of \( v_{\text{norm}}^T \) equals 1.

### A.1.2 Speed and eigenvectors

We have claimed that
\[ s = \sqrt{\frac{e_{13}^2}{1 - e_{13}^2}} \quad \text{T rank 1} \] (A.3)
\[ s = \sqrt{\frac{1 - e_{13}^2}{e_{33}^2}} \quad \text{T rank 2} \] (A.4)

are measures of the magnitude of local velocity.

**Proof:** (A.3) follows from (A.2) since
\[
\|v_{\text{norm}}\|^2 = \frac{e_{13}^2}{(e_{11}^2 + e_{12}^2)^2} (e_{11}^2 + e_{12}^2) = \frac{e_{13}^2}{e_{11}^2 + e_{12}^2} = \frac{e_{13}^2}{1 - e_{13}^2}
\]
Similarly, (A.4) follows from (A.1) since
\[
\|v\|^2 = \frac{1}{e_{33}^2} (e_{31}^2 + e_{32}^2) = \frac{1 - e_{13}^2}{e_{33}^2}
\]

### A.1.3 Speed and \( T_{xy} \)

We have claimed that
\[ \text{Tr}T_{xy} = \lambda_1 (1 - e_{13}^2) \quad \text{T rank 1} \] (A.5)
\[ \det T_{xy} = \lambda_1 \lambda_2 e_{33}^2 \quad \text{T rank 2} \] (A.6)

**Proof:** When \( T \) is of rank 1 the leading \( 2 \times 2 \)-submatrix is
\[
T_{xy} = \lambda_1 \begin{pmatrix}
\frac{e_{11}^2}{e_{11} e_{12}} & e_{11} e_{12} \\
e_{11} e_{12} & \frac{e_{12}^2}{e_{11} e_{12}}
\end{pmatrix}
\]
Figure A.1: Moving point case. Left: Image plane. Right: Spatiotemporal volume.

Figure A.2: Moving line case. Left: Image plane. Right: Spatiotemporal volume.
Since \( ||e_1||^2 = e_{11}^2 + e_{12}^2 + e_{13}^2 = 1 \), (A.5) follows. When \( T \) is of rank 2 the leading 2 \( \times \) 2-submatrix is
\[
T_{xy} = \begin{pmatrix}
\lambda_1 e_{11} + \lambda_2 e_{21} & \lambda_1 e_{12} + \lambda_2 e_{22} \\
\lambda_1 e_{11} e_{12} + \lambda_2 e_{21} e_{22} & \lambda_1 e_{12}^2 + \lambda_2 e_{22}^2
\end{pmatrix}
\]
so that the determinant is
\[
\det T_{xy} = \lambda_1 \lambda_2 [e_{11} e_{22}^2 + e_{12} e_{21}^2 - 2 e_{11} e_{12} e_{21} e_{22}] = \lambda_1 \lambda_2 (e_{11} e_{22} - e_{12} e_{21})^2
\]
But
\[
e_{33} = [e_1 \times e_2]_3 = \begin{vmatrix}
\hat{x} & \hat{y} & \hat{z} \\
e_{11} & e_{12} & e_{13} \\
e_{21} & e_{22} & e_{23}
\end{vmatrix} = e_{11} e_{22} - e_{12} e_{21}
\]
and we are done. \( \square \)

### A.2 Motion models

#### A.2.1 Accuracy of affine parameter estimates

To determine the accuracy of the affine parameter estimates a synthetic 2D profile, Figure A.3, was subjected to sequences of affine transformations. The object covered 100 \( \times \) 100 pixels when not transformed. All experiments started with the object in a standard position, centred in the field of view.

We separately studied expansion (Figure A.5), rotation (Figure A.6), translation (Figure A.7), and composite transformation (Figure A.8). In the figures, histograms of estimated values of the affine parameters are plotted as
\[
\begin{bmatrix}
a_{11} & a_{12} & v_x \\
a_{21} & a_{22} & v_y
\end{bmatrix}
\]
As is seen, the estimates are excellent over a wide range of parameter values. Of course, these estimates have been obtained under perfect conditions and thus merely show that the algorithm then has the ability to produce consistent estimates with small variance.

To see how the algorithm performs in somewhat more realistic situations, a simple camera simulation was generated by adding 10 dB SNR additive uncorrelated noise and motion blurring (moving average). Results are shown in Figures A.9 – A.12. As expected, very little or no degradation in performance can be detected—the spatiotemporal filtering has proven very noise tolerant, with an average orientation estimation error of 3.0° at 10 dB SNR (0.8° at \( \infty \) dB SNR) for the particular filter set used here, [63].
Figure A.3: Textured profile used in experiments.

Figure A.4: A more realistic situation with 10 dB SNR additive uncorrelated noise and image blurring.
Figure A.5: Expansion. (a) $a_{11} = a_{22} = 0.005$. (b) $a_{11} = a_{22} = 0.01$. (c) $a_{11} = a_{22} = 0.02$. (d) $a_{11} = a_{22} = 0.04$. 
Figure A.6: Rotation. (a) $a_{12} = -a_{21} = 0.01$. (b) $a_{12} = -a_{21} = 0.02$. (c) $a_{12} = -a_{21} = 0.04$. (d) $a_{12} = -a_{21} = 0.08$. 
Figure A.7: Translation.  (a) \( v_x = v_y = 0.5 \).  (b) \( v_x = v_y = 1 \).  (c) \( v_x = v_y = 2 \).  (d) \( v_x = v_y = 4 \).

Figure A.8: Composite transformation.  (a) \( a_{11} = a_{12} = -a_{21} = a_{22} = 0.015 \).  \( v_x = -v_y = 1 \).  (b) \( a_{11} = a_{12} = -a_{21} = a_{22} = 0.03 \).  \( v_x = -v_y = 2 \).
Figure A.9: Expansion. (a) $a_{11} = a_{22} = 0.005$. (b) $a_{11} = a_{22} = 0.01$. (c) $a_{11} = a_{22} = 0.02$. (d) $a_{11} = a_{22} = 0.04$. 
Figure A.10: Rotation. (a) $a_{12} = -a_{21} = 0.01$. (b) $a_{12} = -a_{21} = 0.02$. (c) $a_{12} = -a_{21} = 0.04$. (d) $a_{12} = -a_{21} = 0.08$. 
Figure A.11: Translation. (a) $v_x = v_y = 0.5$. (b) $v_x = v_y = 1$. (c) $v_x = v_y = 2$. (d) $v_x = v_y = 4$.

Figure A.12: Composite transformation. (a) $a_{11} = a_{12} = -a_{21} = a_{22} = 0.015$. $v_x = -v_y = 1$. (b) $a_{11} = a_{12} = -a_{21} = a_{22} = 0.03$. $v_x = -v_y = 2$. 
Bibliography


I would like to present a medal of valour to the patient reader who has managed to plough through the volume. Experience shows that a long–term exposure to the material is accompanied by feelings of confusion, frustration, and, finally, resignation.

The author