Linköping Studies in Science and Technology. Dissertations
No. 790

Polynomial Expansion for Orientation and Motion Estimation

Gunnar Farnebäck

INSTITUTE OF TECHNOLOGY
LINKÖPINGS UNIVERSITET

Department of Electrical Engineering
Linköpings universitet, SE-581 83 Linköping, Sweden

Linköping November 2002

Polynomial Expansion for Orientation and Motion Estimation

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping
Sweden

ISBN 91-7373-475-6 ISSN 0345-7524

To Malin

Abstract

This thesis introduces a new signal transform, called polynomial expansion, and based on this develops novel methods for estimation of orientation and motion. The methods are designed exclusively in the spatial domain and can be used for signals of any dimensionality.

Two important concepts in the use of the spatial domain for signal processing is projections into subspaces, e.g. the subspace of second degree polynomials, and representations by frames, e.g. wavelets. It is shown how these concepts can be unified in a least squares framework for representation of finite dimensional vectors by bases, frames, subspace bases, and subspace frames.

This framework is used to give a new derivation of normalized convolution, a method for signal analysis that takes uncertainty in signal values into account and also allows for spatial localization of the analysis functions.

Polynomial expansion is a transformation which at each point transforms the signal into a set of expansion coefficients with respect to a polynomial local signal model. The expansion coefficients are computed using normalized convolution. As a consequence polynomial expansion inherits the mechanism for handling uncertain signals and the spatial localization feature allows good control of the properties of the transform. It is shown how polynomial expansion can be computed very efficiently.

As an application of polynomial expansion, a novel method for estimation of orientation tensors is developed. A new concept for orientation representation, orientation functionals, is introduced and it is shown that orientation tensors can be considered a special case of this representation. By evaluation on a test sequence it is demonstrated that the method performs excellently.

Considering an image sequence as a spatiotemporal volume, velocity can be estimated from the orientations present in the volume. Two novel methods for velocity estimation are presented, with the common idea to combine the orientation tensors over some region for estimation of the velocity field according to a parametric motion model, e.g. affine motion. The first method involves a simultaneous segmentation and velocity estimation algorithm to obtain appropriate regions. The second method is designed for computational efficiency and uses local neighborhoods instead of trying to obtain regions with coherent motion. By evaluation on the Yosemite sequence, it is shown that both methods give substantially more accurate results than previously published methods.

Another application of polynomial expansion is a novel displacement estimation algorithm, i.e. an algorithm which estimates motion from only two consecutive frames rather than from a whole spatiotemporal volume. This approach is necessary when the motion is not temporally coherent, e.g. because the camera is affected by vibrations. It is shown how moving objects can robustly be detected in such image sequences by using the plane+parallax approach to separate out the background motion.

To demonstrate the power of being able to handle uncertain signals it is shown how normalized convolution and polynomial expansion can be computed for interlaced video signals. Together with the displacement estimation algorithm this gives a method to estimate motion from a single interlaced frame.

Acknowledgements

This thesis could never have been written without the contributions from a large number of people. In particular I want to thank the following persons:

My fiancée Malin, for love, support, and patience.

Professor Gösta Granlund, the head of the research group and my supervisor, for showing confidence in my work and letting me pursue my research ideas.

Present and past members of the Computer Vision Laboratory, for providing a stimulating research environment and good friendship.

Björn Johansson, for numerous constructive and interesting discussions, and for proofreading the manuscript.

Dr. Klas Nordberg and Professor Hans Knutsson for many good ideas and discussions.

Visiting Professor Todd Reed and Dr. Hagen Spies for proofreading parts of the manuscript at different times in the production.

Dr. Peter Hackman, Dr. Arne Enqvist, and Dr. Thomas Karlsson at the Department of Mathematics, for skillful teaching of undergraduate mathematics and for consultations on some of the mathematical details in the thesis.

Professor Lars Eldén, also at the Department of Mathematics, for help with some numerical aspects of the thesis.

Johan Wiklund, for keeping the computers happy.

The Knut and Alice Wallenberg foundation, for funding the research within the WITAS project.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Organization . . . 2
- 1.3 Contributions . . . 3
- 1.4 Previous Publications . . . 4
- 1.5 Notation . . . 4

2 A Unified Framework for Bases, Frames, Subspace Bases, and Subspace Frames **7**
- 2.1 Introduction . . . 7
- 2.2 Preliminaries . . . 7
  - 2.2.1 Notation . . . 7
  - 2.2.2 The Linear Equation System . . . 8
  - 2.2.3 The Linear Least Squares Problem . . . 8
  - 2.2.4 The Minimum Norm Problem . . . 8
  - 2.2.5 The Singular Value Decomposition . . . 9
  - 2.2.6 The Pseudo-Inverse . . . 9
  - 2.2.7 The General Linear Least Squares Problem . . . 10
  - 2.2.8 Numerical Aspects . . . 10
- 2.3 Representation by Sets of Vectors . . . 10
  - 2.3.1 Notation . . . 10
  - 2.3.2 Definitions . . . 11
  - 2.3.3 Dual Vector Sets . . . 11
  - 2.3.4 Representation by a Basis . . . 11
  - 2.3.5 Representation by a Frame . . . 12
  - 2.3.6 Representation by a Subspace Basis . . . 12
  - 2.3.7 Representation by a Subspace Frame . . . 12
  - 2.3.8 The Double Dual . . . 13
  - 2.3.9 A Note on Notation . . . 13
- 2.4 Weighted Norms . . . 14
  - 2.4.1 Notation . . . 14
  - 2.4.2 The Weighted General Linear Least Squares Problem . . . 14
  - 2.4.3 Representation by Vector Sets . . . 14
  - 2.4.4 Dual Vector Sets . . . 15

2.5 Weighted Seminorms 16
2.5.1 The Seminorm Weighted General Linear Least Squares Problem 16
2.5.2 Representation by Vector Sets and Dual Vector Sets 17

3 Normalized Convolution 19
3.1 Introduction 19
3.2 Definition of Normalized Convolution 20
3.2.1 Signal and Certainty 20
3.2.2 Basis Functions and Applicability 21
3.2.3 Definition 21
3.2.4 Comments on the Definition 22
3.3 Implementational Issues 23
3.4 Example 24
3.5 Output Certainty 26
3.6 Normalized Differential Convolution 27
3.7 Reduction to Ordinary Convolution 28
3.8 Filter Responses on Uncertain Data 29
3.8.1 Corresponding Basis Functions 29
3.8.2 Practical Considerations 30
3.8.3 Alternative Interpretations 33
3.9 Application Examples 33
3.9.1 Normalized Averaging 33
3.9.2 The Cubic Facet Model 36
3.10 Choosing the Applicability 37
3.11 Towards a Statistical Interpretation of Certainty 37
3.12 Further Generalizations of Normalized Convolution 38

4 Polynomial Expansion 41
4.1 Introduction 41
4.2 Estimating the Coefficients of a Polynomial Model 42
4.3 Fast Implementation 43
4.3.1 Equivalent Correlation Kernels 43
4.3.2 Cartesian Separability 47
4.4 Computational Complexity 49
4.5 Polynomial Expansion in Multiple Scales 52
4.6 Approximative Methods 54
4.6.1 Derivative Filters 54
4.6.2 Binomial Filters 55
4.7 Higher Degree Polynomial Expansion 56
4.8 Relation to Other Approaches 56
4.8.1 Relation to First and Second Derivatives 56
4.8.2 Comparison with Polynomial Facet Models 58
4.8.3 Relation to Polynomial Fit Filters 59

5 Orientation Estimation **61**
5.1 Introduction . . . 61
5.2 The Orientation Tensor . . . 62
5.2.1 Representation of Orientation for Simple Signals . . . 62
5.2.2 Estimation . . . 63
5.2.3 Interpretation for Non-Simple Signals . . . 63
5.3 Orientation Functionals . . . 64
5.4 Tensor Estimation Based on Polynomial Expansion . . . 66
5.4.1 Linear Neighborhoods . . . 66
5.4.2 Quadratic Neighborhoods . . . 66
5.4.3 General Neighborhoods . . . 69
5.5 Properties of the Estimated Tensor . . . 69
5.6 Computational Complexity . . . 71
5.7 Evaluation . . . 72
5.7.1 The Importance of Isotropy . . . 74
5.7.2 Gaussian Applicabilities . . . 77
5.7.3 Choosing γ . . . 79
5.7.4 Best Results . . . 79
5.8 Discussion . . . 80
5.8.1 Multiple Scales . . . 80
5.8.2 Different Radial Functions . . . 81
5.8.3 Additional Basis Functions . . . 81
5.9 Phase Functionals . . . 81

6 Velocity Estimation **83**
6.1 Introduction . . . 83
6.2 From Orientation to Motion . . . 83
6.3 Estimating a Parameterized Velocity Field . . . 84
6.3.1 Motion Models . . . 85
6.3.2 Cost Functions . . . 85
6.3.3 Parameter Estimation . . . 87
6.4 Simultaneous Segmentation and Velocity Estimation . . . 89
6.4.1 The Competitive Algorithm . . . 89
6.4.2 Candidate Regions . . . 90
6.4.3 Segmentation Algorithm . . . 91
6.5 A Fast Velocity Estimation Algorithm . . . 92
6.6 Evaluation . . . 95
6.6.1 Implementation and Performance . . . 97
6.6.2 Results for the Yosemite Sequence . . . 98
6.6.3 Results for the Diverging Tree Sequence . . . 103
6.6.4 Results for Real Image Sequences . . . 103

7 Displacement Estimation **107**
7.1 Introduction . . . 107
7.2 Displacement of a Quadratic Polynomial . . . 107
7.2.1 Intuitive Explanation . . . 110
7.2.2 Practical Considerations . . . 111
7.3 A Simple Disparity Estimation Algorithm . . . 111
7.4 Displacement Estimation . . . 115
7.5 Estimating a Parameterized Displacement Field . . . 116
7.6 Incorporating A Priori Knowledge . . . 116
7.7 Iterative and Multi-scale Displacement Estimation . . . 117
7.7.1 Iterative Displacement Estimation . . . 117
7.7.2 Multi-scale Displacement Estimation . . . 120
7.8 Computational Complexity . . . 120
7.9 Evaluation . . . 122
7.9.1 Global Translation Experiment . . . 122
7.9.2 Results for Yosemite and Diverging Tree . . . 126
7.9.3 Results for Real Image Sequences . . . 127
7.10 Moving Object Detection . . . 127
7.10.1 The Plane+Parallax Decomposition . . . 131
7.10.2 Estimating Background Displacement . . . 131
7.10.3 Reducing Noise . . . 132
7.11 Discussion . . . 132

8 Analysis of Interlaced Signals **137**
8.1 Introduction . . . 137
8.2 The Geometry of Interlaced Signals . . . 137
8.3 Normalized Convolution . . . 138
8.4 Adaptation of Convolution Kernels . . . 141
8.5 Polynomial Expansion . . . 141
8.6 One Frame Motion Estimation . . . 143

9 The Spatial Domain Toolbox **147**
9.1 Introduction . . . 147
9.2 Functions in the Toolbox . . . 147

10 Conclusions and Future Research Directions **149**
10.1 Conclusions . . . 149
10.2 Future Research Directions . . . 150

Appendices **153**
A A Matrix Inversion Lemma . . . 153
B Cartesian Separable and Isotropic Functions . . . 155
C Correlator Structure for Separable Normalized Convolution . . . 158
D Gaussian Correlation Formulas . . . 159
E Binomial Identities . . . 162
F Angular RMS Error . . . 163
G Removing the Isotropic Part of a 3D Tensor . . . 164

H Elementary Stereo Geometry 167
I Correlator Structure for Interlaced Signals 169
J An Adaptive Filtering Approach 170

Chapter 1

Introduction

1.1 Motivation

In this Ph.D. thesis a general framework for spatial domain design of multidimensional signal analysis algorithms is presented. Using this framework, novel algorithms for estimation of orientation, velocity, and displacement are developed. It is more conventional for such methods to be designed, at least partially, in the Fourier domain. To understand why we wish to avoid the use of the Fourier domain altogether, it is necessary to have some background information.

The theory and methods presented in this thesis are results of the research within the WITAS[1] project [99]. The goal of this project is to develop an autonomous flying vehicle and naturally the vision subsystem is an important component. Unfortunately the needed image processing has a tendency to be computationally very demanding and therefore it is of interest to find ways to reduce the amount of processing. One way to do this is to emulate biological vision by using foveally sampled images, i.e. having a higher sampling density in an area of interest and gradually lower sampling density further away. In contrast to the usual rectangular grids, this approach leads to highly irregular sampling patterns.

Except for some very specific sampling patterns, e.g. the logarithmic polar [17, 72, 82, 92], the theory for irregularly sampled multidimensional signals is far less developed than the corresponding theory for regularly sampled signals. Some work has been done on the problem of reconstructing irregularly sampled bandlimited signals [30]. In contrast to the regular case this turns out to be quite complicated, one reason being that the Nyquist frequency varies spatially with the local sampling density. In fact the use of the Fourier domain in general, e.g. for filter design, becomes much more complicated and for this reason we turn our attention to the spatial domain.

As is often the case though, research does not always follow the planned routes. Irregular sampling has its own problems, both with respect to sensor hardware and software implementations of the algorithms. As a result this approach has not been

[1] Wallenberg laboratory for Information Technology for Autonomous Systems.

pursued within the WITAS project.

This is, however, not such a bad failure as it may sound like. The spatial domain approach has turned out to be very fruitful when applied to regularly sampled signals, giving efficient and accurate algorithms.

1.2 Organization

An important concept in the use of the spatial domain for signal processing is projections into subspaces, e.g. the subspace of second degree polynomials. Chapter 2 presents a unified framework for representations of finite dimensional vectors by bases, frames, subspace bases, and subspace frames. The basic idea is that all these representations by sets of vectors can be regarded as solutions to various least squares problems. Generalizations to weighted least squares problems are explored and dual vector sets are derived for efficient computation of the representations.

In chapter 3 the theory developed in the previous chapter is used to derive the method called normalized convolution. This method is a powerful tool for signal analysis in the spatial domain, being able to take uncertainties in the signal values into account and allowing spatial localization of the analysis functions. It is shown how this operation relates to ordinary convolution and how to use normalized convolution to compute filter responses on uncertain data.

Chapter 4 introduces a signal transform called polynomial expansion, which locally projects each neighborhood of the signal onto a polynomial basis. This is done by means of normalized convolution and it is shown that under certain restrictions the polynomial expansion can be computed very efficiently.

Chapter 5 introduces orientation functionals for representation of orientation and it is shown that orientation tensors can be regarded as a special case of this concept. A new method to compute orientation tensors, using the polynomial expansion of the signal, is developed. It is shown by evaluation on a test volume that it performs excellently.

In chapter 6 the orientation tensors from the previous chapter are utilized for velocity estimation. With the idea to estimate velocity over whole regions according to some motion model, two different algorithms are developed. The first one is a simultaneous segmentation and velocity estimation algorithm, while the second one gains in computational efficiency by disregarding the need for a proper segmentation into regions with coherent motion. By evaluation on the Yosemite sequence it is shown that both algorithms are substantially more accurate than previously published methods for velocity estimation.

A different method to estimate motion, in the form of displacement between two signals, is developed in chapter 7. This is also based on the polynomial expansion of the two signals, but works directly on the expansion coefficients instead of using orientation tensors. Combined with the plane+parallax approach it gives a robust method to detect moving objects in aerial scenes.

Chapter 8 discusses how the methods developed in the previous chapters can be adapted for direct use on interlaced video signals, without needing a de-interlacing step. This is the only remnant of the plans to study irregularly sampled signals.

Most algorithms developed in this thesis have been implemented in Matlab. The source of many of them is freely available as a package called The Spatial Domain Toolbox. Chapter 9 gives an overview of this toolbox.

The thesis concludes with chapter 10, summarizing the contents of the thesis and discussing possible directions for future research.

1.3 Contributions

It is never easy to say for sure what ideas and methods are new and which have been published somewhere previously. The following is an attempt at listing the parts of the material that originates with the author and are more or less likely to be truly novel.

The main contribution in chapter 2 is the unification of the seemingly disparate concepts of frames and subspace bases in a least squares framework, together with bases and subspace frames. Other original ideas is the simultaneous weighting in both the signal and coefficient spaces for subspace frames, the full generalization of dual vector sets to the weighted norm case in section 2.4.4, and most of the results in section 2.5 on the weighted seminorm case. The concept of weighted linear combinations in section 2.4.4 may also be novel.

The method of normalized convolution in Chapter 3 is certainly not original work. The primary contribution here is in the presentation of the method. By taking advantage of the framework from chapter 2 to derive the method, the goal is to achieve greater clarity than in earlier presentations. There are also some new contributions to the theory, such as parts of the discussion about output certainty in section 3.5, most of sections 3.8 and 3.10, and all of sections 3.11 and 3.12.

In chapter 4 everything is original except sections 4.6.1, 4.8.2, 4.8.3. The ideas in section 4.5 were developed in close collaboration with Björn Johansson. The main contribution of the chapter is the efficient implementation of the polynomial expansion in section 4.3. The results on separable computation of normalized convolution in section 4.4 are not limited to a polynomial basis but applies to any set of Cartesian separable basis functions and applicabilities. This makes it possible to do the computations significantly more efficient and is obviously an important contribution to the theory of normalized convolution.

In chapter 5 everything is original except section 5.2 about the tensor representation of orientation and estimation of tensors by means of quadrature filter responses. The main contributions are the concept of orientation functionals in section 5.3, the method to estimate orientation tensors from polynomial expansion coefficients in section 5.4, and the observation of the importance of isotropy in section 5.7.1.

Chapter 6 mostly contains original work too, with the exception of sections 6.2 and 6.3.1. The main contributions here are the methods for estimation of motion model parameters in section 6.3, the algorithm for simultaneous segmentation and velocity estimation in section 6.4, and the fast velocity estimation algorithm in section 6.5.

All material in both chapters 7 and 8 is original, except the plane+parallax

approach in section 7.10.1 and the interlaced geometry in section 8.2.

1.4 Previous Publications

Large parts of the material have previously been published in earlier thesis work and at conferences, as listed below.

[21]	Master's thesis	Most of the material in sections 6.2–6.4, although with a different focus.
[23]	Licentiate thesis[2]	Central parts of chapters 2–6. There are many revisions and expansions of the material in this thesis though.
[22]	SSAB 1997	A condensed version of my master's thesis.
[24]	SCIA 1999	Chapter 2 and minor pieces from chapter 3.
[25]	ICPR 2000	Chapter 6 except section 6.4. This paper won an Honorable Mention in the Best Student Paper Award.
[26]	VMV 2000	Large parts of chapters 4 and 5.
[27]	SSAB 2001	Sections 7.2 and 7.3.
[28]	ICCV 2001	Chapter 6 with emphasis on section 6.4.
[29]	SSAB 2002	Most of chapter 7 except section 7.3.

Additionally, the velocity estimation algorithm in section 6.5 participated in the Algorithm Performance Contest at ICPR 2000 [2].

1.5 Notation

Lowercase letters in boldface ($\mathbf{v}$) are used for vectors and in matrix algebra contexts they are always column vectors. Uppercase letters in boldface ($\mathbf{A}$) are used for matrices. The conjugate transpose of a matrix or a vector is denoted $\mathbf{A}^*$. The transpose of a real matrix or vector is also denoted $\mathbf{A}^T$. Complex conjugation without transpose is denoted $\bar{\mathbf{v}}$. The standard inner product between two vectors is written $(\mathbf{u}, \mathbf{v})$ or $\mathbf{u}^*\mathbf{v}$. The norm of a vector is induced from the inner product,

$$\|\mathbf{v}\| = \sqrt{\mathbf{v}^*\mathbf{v}}. \tag{1.1}$$

Weighted inner products are given by

$$(\mathbf{u}, \mathbf{v})_{\mathbf{W}} = (\mathbf{W}\mathbf{u}, \mathbf{W}\mathbf{v}) = \mathbf{u}^*\mathbf{W}^*\mathbf{W}\mathbf{v} \tag{1.2}$$

and the induced weighted norms by

$$\|\mathbf{v}\|_{\mathbf{W}} = \sqrt{(\mathbf{v}, \mathbf{v})_{\mathbf{W}}} = \sqrt{(\mathbf{W}\mathbf{v}, \mathbf{W}\mathbf{v})} = \|\mathbf{W}\mathbf{v}\|, \tag{1.3}$$

where $\mathbf{W}$ normally is a positive definite Hermitian matrix. In the case that it is only positive semidefinite we instead have a weighted seminorm. The norm of a

[2] A licentiate is an intermediate degree between master and doctor.

matrix is assumed to be the Frobenius norm, $\|\mathbf{A}\|^2 = \mathrm{tr}\,(\mathbf{A}^*\mathbf{A})$, where the trace of a quadratic matrix, $\mathrm{tr}\,\mathbf{M}$, is the sum of the diagonal elements. The pseudo-inverse of a matrix is denoted $\mathbf{A}^\dagger$. Somewhat nonstandard is the use of $\mathbf{u} \cdot \mathbf{v}$ to denote pointwise multiplication of the elements of two vectors. Furthermore $\hat{\mathbf{v}}$ is used to denote vectors of unit length and $\tilde{\mathbf{v}}$ is used for dual vectors. Binomial coefficients are denoted $\binom{n}{k}$. Additional notation is introduced where needed, e.g. $f \star g$ to denote unnormalized cross correlation in section 3.7.

Chapter 2

A Unified Framework for Bases, Frames, Subspace Bases, and Subspace Frames

2.1 Introduction

Frames and subspace bases, and of course bases, are well known concepts, which have been covered in several publications. Usually, however, they are treated as disparate entities. The idea behind this presentation of the material is to give a unified framework for bases, frames, and subspace bases, as well as the somewhat less known subspace frames. The basic idea is that the coefficients in the representation of a vector in terms of a frame, etc., can be described as solutions to various least squares problems. Using this to define what coefficients should be used, expressions for dual vector sets are derived. These results are then generalized to the case of weighted norms and finally also to the case of weighted seminorms. The presentation is restricted to finite dimensional vector spaces and relies heavily on matrix representations.

2.2 Preliminaries

To begin with, we review some basic concepts from (numerical) linear algebra. All of these results are well known and can be found in any modern textbook on numerical linear algebra, e.g. [35].

2.2.1 Notation

Let $\mathcal{C}^n$ be an n-dimensional complex vector space. Elements of this space are denoted by lower-case bold letters, e.g. $\mathbf{v}$, indicating $n \times 1$ column vectors. Upper-case bold letters, e.g. $\mathbf{F}$, denote complex matrices. With $\mathcal{C}^n$ is associated the

standard inner product, $(\mathbf{f}, \mathbf{g}) = \mathbf{f}^*\mathbf{g}$, where * denotes conjugate transpose, and the Euclidian norm, $\|f\| = \sqrt{(\mathbf{f}, \mathbf{f})}$.

In this section $\mathbf{A}$ is an $n \times m$ complex matrix, $\mathbf{b} \in \mathcal{C}^n$, and $\mathbf{x} \in \mathcal{C}^m$.

2.2.2 The Linear Equation System

The linear equation system

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.1}$$

has a unique solution

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{2.2}$$

if and only if $\mathbf{A}$ is square and non-singular. If the equation system is overdetermined it does in general not have a solution and if it is underdetermined there are normally an infinite set of solutions. In these cases the equation system can be solved in a least squares and/or minimum norm sense, as discussed below.

2.2.3 The Linear Least Squares Problem

Assume that $n \geq m$ and that $\mathbf{A}$ is of rank m (full column rank). Then the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ is not guaranteed to have a solution and the best we can do is to minimize the residual error.

The linear least squares problem

$$\arg\min_{\mathbf{x} \in \mathcal{C}^n} \|\mathbf{A}\mathbf{x} - \mathbf{b}\| \tag{2.3}$$

has the unique solution

$$\mathbf{x} = (\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^*\mathbf{b}. \tag{2.4}$$

If $\mathbf{A}$ is rank deficient the solution is not unique, a case which we return to in section 2.2.7.

2.2.4 The Minimum Norm Problem

Assume that $n \leq m$ and that $\mathbf{A}$ is of rank n (full row rank). Then the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ may have more than one solution and to choose between them we take the one with minimum norm.

The minimum norm problem

$$\arg\min_{\mathbf{x} \in \mathcal{S}} \|\mathbf{x}\|, \qquad \mathcal{S} = \{\mathbf{x} \in \mathcal{C}^m; \mathbf{A}\mathbf{x} = \mathbf{b}\} \tag{2.5}$$

has the unique solution

$$\mathbf{x} = \mathbf{A}^*(\mathbf{A}\mathbf{A}^*)^{-1}\mathbf{b}. \tag{2.6}$$

If $\mathbf{A}$ is rank deficient it is possible that there is no solution at all, a case to which we return in section 2.2.7.

2.2.5 The Singular Value Decomposition

An arbitrary matrix $\mathbf{A}$ of rank r can be factorized by the Singular Value Decomposition, SVD, as

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*, \tag{2.7}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, $n \times n$ and $m \times m$ respectively. $\mathbf{\Sigma}$ is a diagonal $n \times m$ matrix

$$\mathbf{\Sigma} = \text{diag}\left(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0\right), \tag{2.8}$$

where $\sigma_1, \ldots, \sigma_r$ are the non-zero singular values. The singular values are all real and $\sigma_1 \geq \ldots \geq \sigma_r > 0$. If $\mathbf{A}$ is of full rank we have $r = \min(n, m)$ and all singular values are non-zero.

2.2.6 The Pseudo-Inverse

The pseudo-inverse[1] $\mathbf{A}^\dagger$ of any matrix $\mathbf{A}$ can be defined via the SVD given by (2.7) and (2.8) as

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^*, \tag{2.9}$$

where $\mathbf{\Sigma}^\dagger$ is a diagonal $m \times n$ matrix

$$\mathbf{\Sigma}^\dagger = \text{diag}\left(\tfrac{1}{\sigma_1}, \ldots, \tfrac{1}{\sigma_r}, 0, \ldots, 0\right). \tag{2.10}$$

We can notice that if $\mathbf{A}$ is of full rank and $n \geq m$, then the pseudo-inverse can also be computed as

$$\mathbf{A}^\dagger = (\mathbf{A}^*\mathbf{A})^{-1}\mathbf{A}^* \tag{2.11}$$

and if instead $n \leq m$ then

$$\mathbf{A}^\dagger = \mathbf{A}^*(\mathbf{A}\mathbf{A}^*)^{-1}. \tag{2.12}$$

If $m = n$ then $\mathbf{A}$ is quadratic and the condition of full rank becomes equivalent with non-singularity. It is obvious that both the equations (2.11) and (2.12) reduce to

$$\mathbf{A}^\dagger = \mathbf{A}^{-1} \tag{2.13}$$

in this case.

Regardless of rank conditions we have the following useful identities,

$$(\mathbf{A}^\dagger)^\dagger = \mathbf{A}, \tag{2.14}$$

$$(\mathbf{A}^*)^\dagger = (\mathbf{A}^\dagger)^*, \tag{2.15}$$

$$\mathbf{A}^\dagger = (\mathbf{A}^*\mathbf{A})^\dagger\mathbf{A}^*, \tag{2.16}$$

$$\mathbf{A}^\dagger = \mathbf{A}^*(\mathbf{A}\mathbf{A}^*)^\dagger. \tag{2.17}$$

[1] This pseudo-inverse is also known as the Moore-Penrose inverse.

2.2.7 The General Linear Least Squares Problem

The remaining case is when $\mathbf{A}$ is rank deficient. Then the equation $\mathbf{Ax} = \mathbf{b}$ is not guaranteed to have a solution and there may be more than one $\mathbf{x}$ minimizing the residual error. This problem can be solved as a simultaneous least squares and minimum norm problem.

The general (or rank deficient) linear least squares problem is stated as

$$\arg\min_{\mathbf{x}\in\mathcal{S}} \|\mathbf{x}\|, \qquad \mathcal{S} = \{\mathbf{x} \in \mathcal{C}^m; \|\mathbf{Ax} - \mathbf{b}\| \text{ is minimum}\}, \tag{2.18}$$

i.e. among the least squares solutions, choose the one with minimum norm. Clearly this formulation contains both the ordinary linear least squares problem and the minimum norm problem as special cases. The unique solution is given in terms of the pseudo-inverse as

$$\mathbf{x} = \mathbf{A}^\dagger \mathbf{b}. \tag{2.19}$$

Notice that by equations (2.11)–(2.13) this solution is consistent with (2.2), (2.4), and (2.6).

2.2.8 Numerical Aspects

Although the above results are most commonly found in books on *numerical* linear algebra, only their algebraic properties are being discussed here. It should, however, be mentioned that e.g. equations (2.9) and (2.11) have numerical properties that differ significantly. The interested reader is referred to [11].

2.3 Representation by Sets of Vectors

If we have a set of vectors $\{\mathbf{f}_k\} \subset \mathcal{C}^n$ and wish to represent[2] an arbitrary vector $\mathbf{v}$ as a linear combination

$$\mathbf{v} \sim \sum c_k \mathbf{f}_k \tag{2.20}$$

of the given set, how should the coefficients $\{c_k\}$ be chosen? In general this question can be answered in terms of linear least squares problems.

2.3.1 Notation

With the set of vectors, $\{\mathbf{f}_k\}_{k=1}^m \subset \mathcal{C}^n$, is associated an $n \times m$ matrix

$$\mathbf{F} = [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_m], \tag{2.21}$$

which effectively is a *reconstructing* operator because multiplication with an $m \times 1$ vector $\mathbf{c}$, $\mathbf{Fc}$, produces linear combinations of the vectors $\{\mathbf{f}_k\}$. In terms of the reconstruction matrix, equation (2.20) can be rewritten as

$$\mathbf{v} \sim \mathbf{Fc}, \tag{2.22}$$

[2]Ideally we would like to have equality in equation (2.20) but that cannot always be obtained.

		spans $\mathcal{C}^n$	
		yes	no
linearly	independent	basis	subspace basis
	dependent	frame	subspace frame

Table 2.1: Definitions

where the coefficients $\{c_k\}$ have been collected in the vector $\mathbf{c}$.

The conjugate transpose of the reconstruction matrix, $\mathbf{F}^*$, gives an analyzing operator because $\mathbf{F}^*\mathbf{x}$ yields a vector containing the inner products between $\{\mathbf{f}_k\}$ and the vector $\mathbf{x} \in \mathcal{C}^n$.

2.3.2 Definitions

Let $\{\mathbf{f}_k\}$ be a subset of $\mathcal{C}^n$. If $\{\mathbf{f}_k\}$ spans $\mathcal{C}^n$ and is linearly independent it is called a *basis*. If it spans $\mathcal{C}^n$ but is linearly dependent it is called a *frame*. If it is linearly independent but does not span $\mathcal{C}^n$ it is called a *subspace basis*. Finally, if it neither spans $\mathcal{C}^n$, nor is linearly independent, it is called a *subspace frame*.[3] This relationship is depicted in table 2.1. If the properties of $\{\mathbf{f}_k\}$ are unknown or arbitrary we simply use *set of vectors* or *vector set* as a collective term.

2.3.3 Dual Vector Sets

We associate with a given vector set $\{\mathbf{f}_k\}$ the dual vector set $\{\tilde{\mathbf{f}}_k\}$, characterized by the condition that for an arbitrary vector $\mathbf{v}$ the coefficients $\{c_k\}$ in equation (2.20) are given as inner products between the dual vectors and $\mathbf{v}$,

$$c_k = (\tilde{\mathbf{f}}_k, \mathbf{v}) = \tilde{\mathbf{f}}_k^* \mathbf{v}. \tag{2.23}$$

This equation can be rewritten in terms of the reconstruction matrix $\tilde{\mathbf{F}}$ corresponding to $\{\tilde{\mathbf{f}}_k\}$ as

$$\mathbf{c} = \tilde{\mathbf{F}}^* \mathbf{v}. \tag{2.24}$$

The existence of the dual vector set is a nontrivial fact, which will be proved in the following sections for the various classes of vector sets.

2.3.4 Representation by a Basis

Let $\{\mathbf{f}_k\}$ be a basis. An arbitrary vector $\mathbf{v}$ can be written as a linear combination of the basis vectors, $\mathbf{v} = \mathbf{F}\mathbf{c}$, for a unique coefficient vector $\mathbf{c}$.[4]

Because $\mathbf{F}$ is invertible in the case of a basis, we immediately get

$$\mathbf{c} = \mathbf{F}^{-1}\mathbf{v} \tag{2.25}$$

[3] The notation used here is somewhat nonstandard. See section 2.3.9 for a discussion.

[4] The coefficients $\{c_k\}$ are of course also known as the *coordinates* for $\mathbf{v}$ with respect to the basis $\{\mathbf{f}_k\}$.

and it is clear from comparison with equation (2.24) that $\tilde{\mathbf{F}}$ exists and is given by

$$\tilde{\mathbf{F}} = (\mathbf{F}^{-1})^*. \tag{2.26}$$

In this very ideal case where the vector set is a basis, there is no need to state a least squares problem to find $\mathbf{c}$ or $\tilde{\mathbf{F}}$. That this could indeed be done is discussed in section 2.3.7.

2.3.5 Representation by a Frame

Let $\{\mathbf{f}_k\}$ be a frame. Because the frame spans $\mathcal{C}^n$, an arbitrary vector $\mathbf{v}$ can still be written as a linear combination of the frame vectors, $\mathbf{v} = \mathbf{F}\mathbf{c}$. This time, however, there are infinitely many coefficient vectors $\mathbf{c}$ satisfying the relation. To get a uniquely determined solution we add the requirement that $\mathbf{c}$ be of minimum norm. This is nothing but the minimum norm problem of section 2.2.4 and equation (2.6) gives the solution

$$\mathbf{c} = \mathbf{F}^*(\mathbf{F}\mathbf{F}^*)^{-1}\mathbf{v}. \tag{2.27}$$

Hence the dual frame exists and is given by

$$\tilde{\mathbf{F}} = (\mathbf{F}\mathbf{F}^*)^{-1}\mathbf{F}. \tag{2.28}$$

2.3.6 Representation by a Subspace Basis

Let $\{\mathbf{f}_k\}$ be a subspace basis. In general, an arbitrary vector $\mathbf{v}$ cannot be written as a linear combination of the subspace basis vectors, $\mathbf{v} = \mathbf{F}\mathbf{c}$. Equality only holds for vectors $\mathbf{v}$ in the subspace spanned by $\{\mathbf{f}_k\}$. Thus we have to settle for the $\mathbf{c}$ giving the closest vector $\mathbf{v}' = \mathbf{F}\mathbf{c}$ in the subspace. Since the subspace basis vectors are linearly independent we have the linear least squares problem of section 2.2.3 with the solution given by equation (2.4) as

$$\mathbf{c} = (\mathbf{F}^*\mathbf{F})^{-1}\mathbf{F}^*\mathbf{v}. \tag{2.29}$$

Hence the dual subspace basis exists and is given by

$$\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{F}^*\mathbf{F})^{-1}. \tag{2.30}$$

Geometrically $\mathbf{v}'$ is the orthogonal projection of $\mathbf{v}$ onto the subspace.

2.3.7 Representation by a Subspace Frame

Let $\{\mathbf{f}_k\}$ be a subspace frame. In general, an arbitrary vector $\mathbf{v}$ cannot be written as a linear combination of the subspace frame vectors, $\mathbf{v} = \mathbf{F}\mathbf{c}$. Equality only holds for vectors $\mathbf{v}$ in the subspace spanned by $\{\mathbf{f}_k\}$. Thus we have to settle for the $\mathbf{c}$ giving the closest vector $\mathbf{v}' = \mathbf{F}\mathbf{c}$ in the subspace. Since the subspace frame vectors are linearly dependent there are also infinitely many $\mathbf{c}$ giving the same closest vector $\mathbf{v}'$, so to distinguish between these we choose the one with minimum

norm. This is the general linear least squares problem of section 2.2.7 with the solution given by equation (2.19) as

$$\mathbf{c} = \mathbf{F}^{\dagger}\mathbf{v}. \tag{2.31}$$

Hence the dual subspace frame exists and is given by

$$\tilde{\mathbf{F}} = (\mathbf{F}^{\dagger})^{*}. \tag{2.32}$$

The subspace frame case is the most general case since all the other ones can be considered as special cases. The only thing that happens to the general linear least squares problem formulated here is that sometimes there is an exact solution $\mathbf{v} = \mathbf{F}\mathbf{c}$, rendering the minimum residual error requirement superfluous, and sometimes there is a unique solution $\mathbf{c}$, rendering the minimum norm requirement superfluous. Consequently the solution given by equation (2.32) subsumes all the other ones, which is in agreement with equations (2.11)–(2.13).

2.3.8 The Double Dual

The dual of $\{\tilde{\mathbf{f}}_k\}$ can be computed from equation (2.32), applied twice, together with (2.14) and (2.15),

$$\tilde{\tilde{\mathbf{F}}} = \tilde{\mathbf{F}}^{\dagger *} = \mathbf{F}^{\dagger * \dagger *} = \mathbf{F}^{\dagger * * \dagger} = \mathbf{F}^{\dagger\dagger} = \mathbf{F}. \tag{2.33}$$

What this means is that if we know the inner products between $\mathbf{v}$ and $\{\mathbf{f}_k\}$ we can reconstruct $\mathbf{v}$ using the *dual* vectors. To summarize we have the two relations

$$\mathbf{v} \sim \mathbf{F}(\tilde{\mathbf{F}}^{*}\mathbf{v}) = \sum_k (\tilde{\mathbf{f}}_k, \mathbf{v})\mathbf{f}_k, \tag{2.34}$$

$$\mathbf{v} \sim \tilde{\mathbf{F}}(\mathbf{F}^{*}\mathbf{v}) = \sum_k (\mathbf{f}_k, \mathbf{v})\tilde{\mathbf{f}}_k. \tag{2.35}$$

2.3.9 A Note on Notation

Usually a frame is defined by the frame condition,

$$A\|\mathbf{v}\|^2 \leq \sum_k |(\mathbf{f}_k, \mathbf{v})|^2 \leq B\|\mathbf{v}\|^2, \tag{2.36}$$

which must hold for some $A > 0$, some $B < \infty$, and all $\mathbf{v} \in \mathcal{C}^n$. In the finite dimensional setting used here the first inequality holds if and only if $\{\mathbf{f}_k\}$ spans all of $\mathcal{C}^n$ and the second inequality is a triviality as soon as the number of frame vectors is finite.

The difference between this definition and the one used in section 2.3.2 is that the bases are included in the set of frames. As we have seen that equation (2.28) is consistent with equation (2.26), the same convention could have been used here. The reason for not doing so is that the presentation would have become more involved.

Likewise, we may allow the subspace bases to span the whole $\mathcal{C}^n$, making bases a special case. Indeed, as has already been discussed to some extent, if subspace frames are allowed to be linearly independent, and/or span the whole $\mathcal{C}^n$, all the other cases can be considered special cases of subspace frames.

2.4 Weighted Norms

An interesting generalization of the theory developed so far is to exchange the Euclidian norms used in all minimizations for weighted norms.

2.4.1 Notation

Let the weighting matrix $\mathbf{W}$ be an $n \times n$ positive definite Hermitian matrix. The weighted inner product $(\cdot,\cdot)_{\mathbf{W}}$ on $\mathcal{C}^n$ is defined by

$$(\mathbf{f},\mathbf{g})_{\mathbf{W}} = (\mathbf{W}\mathbf{f},\mathbf{W}\mathbf{g}) = \mathbf{f}^*\mathbf{W}^*\mathbf{W}\mathbf{g} = \mathbf{f}^*\mathbf{W}^2\mathbf{g} \tag{2.37}$$

and the induced weighted norm $\|\cdot\|_{\mathbf{W}}$ is given by

$$\|\mathbf{f}\|_{\mathbf{W}} = \sqrt{(\mathbf{f},\mathbf{f})_{\mathbf{W}}} = \sqrt{(\mathbf{W}\mathbf{f},\mathbf{W}\mathbf{f})} = \|\mathbf{W}\mathbf{f}\|. \tag{2.38}$$

In this section $\mathbf{M}$ and $\mathbf{L}$ denote weighting matrices for $\mathcal{C}^n$ and $\mathcal{C}^m$ respectively. The notation from previous sections carry over unchanged.

2.4.2 The Weighted General Linear Least Squares Problem

The weighted version of the general linear least squares problem is stated as

$$\arg\min_{\mathbf{x}\in\mathcal{S}} \|\mathbf{x}\|_{\mathbf{L}}, \qquad \mathcal{S} = \{\mathbf{x}\in\mathcal{C}^m; \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_{\mathbf{M}} \text{ is minimum}\}. \tag{2.39}$$

This problem can be reduced to its unweighted counterpart by introducing $\mathbf{x}' = \mathbf{L}\mathbf{x}$, whereby equation (2.39) can be rewritten as

$$\arg\min_{\mathbf{x}'\in\mathcal{S}} \|\mathbf{x}'\|, \qquad \mathcal{S} = \{\mathbf{x}'\in\mathcal{C}^m; \|\mathbf{M}\mathbf{A}\mathbf{L}^{-1}\mathbf{x}'-\mathbf{M}\mathbf{b}\| \text{ is minimum}\}. \tag{2.40}$$

The solution is given by equation (2.19) as

$$\mathbf{x}' = (\mathbf{M}\mathbf{A}\mathbf{L}^{-1})^{\dagger}\mathbf{M}\mathbf{b}, \tag{2.41}$$

which after back-substitution yields

$$\mathbf{x} = \mathbf{L}^{-1}(\mathbf{M}\mathbf{A}\mathbf{L}^{-1})^{\dagger}\mathbf{M}\mathbf{b}. \tag{2.42}$$

2.4.3 Representation by Vector Sets

Let $\{\mathbf{f}_k\} \subset \mathcal{C}^n$ be any type of vector set. We want to represent an arbitrary vector $v \in \mathcal{C}^n$ as a linear combination of the given vectors,

$$\mathbf{v} \sim \mathbf{F}\mathbf{c}, \tag{2.43}$$

where the coefficient vector $\mathbf{c}$ is chosen so that

1. the distance between $\mathbf{v}' = \mathbf{F}\mathbf{c}$ and $\mathbf{v}$, $\|\mathbf{v}' - \mathbf{v}\|_{\mathbf{M}}$, is smallest possible, and
2. the length of $\mathbf{c}$, $\|\mathbf{c}\|_{\mathbf{L}}$, is minimized.

This is of course the weighted general linear least squares problem of the previous section, with the solution

$$\mathbf{c} = \mathbf{L}^{-1}(\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{\dagger}\mathbf{M}\mathbf{v}. \tag{2.44}$$

From the geometry of the problem one would suspect that $\mathbf{M}$ should not influence the solution in the case of a basis or a frame, because the vectors span the whole space so that $\mathbf{v}'$ equals $\mathbf{v}$ and the distance is zero, regardless of norm. Likewise $\mathbf{L}$ should not influence the solution in the case of a basis or a subspace basis. That this is correct can easily be seen by applying the identities (2.11)–(2.13) to the solution (2.44). In the case of a frame we get

$$\begin{aligned}\mathbf{c} &= \mathbf{L}^{-1}(\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{*}((\mathbf{M}\mathbf{F}\mathbf{L}^{-1})(\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{*})^{-1}\mathbf{M}\mathbf{v} \\ &= \mathbf{L}^{-2}\mathbf{F}^{*}\mathbf{M}(\mathbf{M}\mathbf{F}\mathbf{L}^{-2}\mathbf{F}^{*}\mathbf{M})^{-1}\mathbf{M}\mathbf{v} \\ &= \mathbf{L}^{-2}\mathbf{F}^{*}(\mathbf{F}\mathbf{L}^{-2}\mathbf{F}^{*})^{-1}\mathbf{v},\end{aligned} \tag{2.45}$$

in the case of a subspace basis

$$\begin{aligned}\mathbf{c} &= \mathbf{L}^{-1}((\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{*}(\mathbf{M}\mathbf{F}\mathbf{L}^{-1}))^{-1}(\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{*}\mathbf{M}\mathbf{v} \\ &= \mathbf{L}^{-1}(\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{F}\mathbf{L}^{-1})^{-1}\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{v} \\ &= (\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{F})^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{v},\end{aligned} \tag{2.46}$$

and in the case of a basis

$$\mathbf{c} = \mathbf{L}^{-1}(\mathbf{M}\mathbf{F}\mathbf{L}^{-1})^{-1}\mathbf{M}\mathbf{v} = \mathbf{F}^{-1}\mathbf{v}. \tag{2.47}$$

2.4.4 Dual Vector Sets

It is not completely obvious how the concept of a dual vector set should be generalized to the weighted norm case. We would like to retain the symmetry relation from equation (2.33) and get correspondences to the representations (2.34) and (2.35). This can be accomplished by the weighted dual[5]

$$\tilde{\mathbf{F}} = \mathbf{M}^{-1}(\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M})^{\dagger}\mathbf{L}, \tag{2.48}$$

which obeys the relations

$$\tilde{\tilde{\mathbf{F}}} = \mathbf{F}, \tag{2.49}$$

$$\mathbf{v} \sim \mathbf{F}\mathbf{L}^{-2}\tilde{\mathbf{F}}^{*}\mathbf{M}^{2}\mathbf{v}, \tag{2.50}$$

$$\mathbf{v} \sim \tilde{\mathbf{F}}\mathbf{L}^{-2}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{v}. \tag{2.51}$$

[5]To be more precise we should say ML-weighted dual, denoted $\tilde{\mathbf{F}}_{\mathbf{ML}}$. In the current context the extra index would only weigh down the notation, and has therefore been dropped.

Unfortunately the two latter relations are not as easily interpreted as (2.34) and (2.35). The situation simplifies a lot in the special case where $\mathbf{L} = \mathbf{I}$. Then we have

$$\tilde{\mathbf{F}} = \mathbf{M}^{-1}(\mathbf{F}^*\mathbf{M})^\dagger, \tag{2.52}$$

which can be rewritten by identity (2.17) as

$$\tilde{\mathbf{F}} = \mathbf{F}(\mathbf{F}^*\mathbf{M}^2\mathbf{F})^\dagger. \tag{2.53}$$

The two relations (2.50) and (2.51) can now be rewritten as

$$\mathbf{v} \sim \mathbf{F}(\tilde{\mathbf{F}}^*\mathbf{M}^2\mathbf{v}) = \sum_k (\tilde{\mathbf{f}}_k, \mathbf{v})_\mathbf{M}\, \mathbf{f}_k, \tag{2.54}$$

$$\mathbf{v} \sim \tilde{\mathbf{F}}(\mathbf{F}^*\mathbf{M}^2\mathbf{v}) = \sum_k (\mathbf{f}_k, \mathbf{v})_\mathbf{M}\, \tilde{\mathbf{f}}_k. \tag{2.55}$$

Returning to the case of a general $\mathbf{L}$, the factor $\mathbf{L}^{-2}$ in (2.50) and (2.51) should be interpreted as a weighted linear combination, i.e. $\mathbf{F}\mathbf{L}^{-2}\mathbf{c}$ would be an $\mathbf{L}^{-1}$-weighted linear combination of the vectors $\{\mathbf{f}_k\}$, with the coefficients given by $\mathbf{c}$, analogously to $\mathbf{F}^*\mathbf{M}^2\mathbf{v}$ being the set of $\mathbf{M}$-weighted inner products between $\{\mathbf{f}_k\}$ and a vector $\mathbf{v}$.

2.5 Weighted Seminorms

The final level of generalization to be addressed here is when the weighting matrices are allowed to be semidefinite, turning the norms into seminorms. This has fundamental consequences for the geometry of the problem. The primary difference is that with a (proper) seminorm not only the vector $\mathbf{0}$ has length zero, but a whole subspace has. This fact has to be taken into account with respect to the terms spanning and linear dependence.[6]

2.5.1 The Seminorm Weighted General Linear Least Squares Problem

When $\mathbf{M}$ and $\mathbf{L}$ are allowed to be semidefinite[7] the solution to equation (2.39) is given by Eldén in [20] as

$$\mathbf{x} = (\mathbf{I} - (\mathbf{L}\mathbf{P})^\dagger\mathbf{L})(\mathbf{M}\mathbf{A})^\dagger\mathbf{M}\mathbf{b} + \mathbf{P}(\mathbf{I} - (\mathbf{L}\mathbf{P})^\dagger\mathbf{L}\mathbf{P})\mathbf{z}, \tag{2.56}$$

where $\mathbf{z}$ is arbitrary and $\mathbf{P}$ is the projection

$$\mathbf{P} = \mathbf{I} - (\mathbf{M}\mathbf{A})^\dagger\mathbf{M}\mathbf{A}. \tag{2.57}$$

[6] Specifically, if a set of otherwise linearly independent vectors have a linear combination of norm zero, we say that they are *effectively* linearly dependent, since they for all practical purposes may as well have been.

[7] $\mathbf{M}$ and $\mathbf{L}$ may in fact be completely arbitrary matrices of compatible sizes.

Furthermore the solution is unique if and only if

$$\mathcal{N}(\mathbf{MA}) \cap \mathcal{N}(\mathbf{L}) = \{\mathbf{0}\}, \tag{2.58}$$

where $\mathcal{N}(\cdot)$ denotes the null space. When there are multiple solutions, the first term of (2.56) gives the solution with minimum *Euclidian* norm.

If we make the restriction that only $\mathbf{M}$ may be semidefinite, the derivation in section 2.4.2 still holds and the solution is unique and given by equation (2.42) as

$$\mathbf{x} = \mathbf{L}^{-1}(\mathbf{MAL}^{-1})^{\dagger}\mathbf{Mb}. \tag{2.59}$$

2.5.2 Representation by Vector Sets and Dual Vector Sets

Here we have exactly the same representation problem as in section 2.4.3, except that that $\mathbf{M}$ and $\mathbf{L}$ may now be semidefinite. The consequence of $\mathbf{M}$ being semidefinite is that residual errors along some directions do not matter, while $\mathbf{L}$ being semidefinite means that certain linear combinations of the available vectors can be used for free. When both are semidefinite it may happen that some linear combinations can be used freely without affecting the residual error. This causes an ambiguity in the choice of the coefficients $\mathbf{c}$, which can be resolved by the additional requirement that among the solutions, $\mathbf{c}$ is chosen with minimum Euclidian norm. Then the solution is given by the first part of equation (2.56) as

$$\mathbf{c} = (\mathbf{I} - (\mathbf{L}(\mathbf{I} - (\mathbf{MF})^{\dagger}\mathbf{MF}))^{\dagger}\mathbf{L})(\mathbf{MF})^{\dagger}\mathbf{Mv}. \tag{2.60}$$

Since this expression is something of a mess we are not going explore the possibilities of finding a dual vector set or analogues of the relations (2.50) and (2.51). Let us instead turn to the considerably simpler case where only $\mathbf{M}$ is allowed to be semidefinite. As noted in the previous section, we can now use the same solution as in the case with weighted norms, reducing the solution (2.60) to that given by equation (2.44),

$$\mathbf{c} = \mathbf{L}^{-1}(\mathbf{MFL}^{-1})^{\dagger}\mathbf{Mv}. \tag{2.61}$$

Unfortunately we can no longer define the dual vector set by means of equation (2.48), due to the occurrence of an explicit inverse of $\mathbf{M}$. Applying identity (2.16) to (2.61), however, we get

$$\mathbf{c} = \mathbf{L}^{-1}(\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{FL}^{-1})^{\dagger}\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{v} \tag{2.62}$$

and it follows that

$$\tilde{\mathbf{F}} = \mathbf{FL}^{-1}(\mathbf{L}^{-1}\mathbf{F}^{*}\mathbf{M}^{2}\mathbf{FL}^{-1})^{\dagger}\mathbf{L} \tag{2.63}$$

yields a dual satisfying the relations (2.49)–(2.51). In the case that $\mathbf{L} = \mathbf{I}$ this expression simplifies further to (2.53), just as for weighted norms. For future reference we also notice that (2.61) reduces to

$$\mathbf{c} = (\mathbf{MF})^{\dagger}\mathbf{Mv}. \tag{2.64}$$

Chapter 3

Normalized Convolution

3.1 Introduction

Normalized convolution is a method for signal analysis that takes uncertainties in signal values into account and at the same time allows spatial localization of possibly unlimited analysis functions. The method was primarily developed by Knutsson and Westin [64, 66, 94] and has later been described and/or used in e.g. [3, 4, 5, 40, 54, 56, 57, 61, 84, 85, 86, 93]. The conceptual basis for the method is the signal/certainty philosophy [38, 39, 62, 97], i.e. separating the values of a signal from the certainty of the measurements. Some of the ideas used in normalized convolution can also be found in [16], further discussed in section 4.8.3.

Most of the previous presentations of normalized convolution have primarily been set in a tensor algebra framework, with only some mention of the relations to least squares problems. Here we will skip the tensor algebra approach completely and instead use the framework developed in chapter 2 as the theoretical basis for deriving normalized convolution. Specifically, we will use the theory of subspace bases and the connections to least squares problems. Readers interested in the tensor algebra approach are referred to [64, 66, 93, 94].

Normalized convolution can, for each neighborhood of the signal, geometrically be interpreted as a projection into a subspace which is spanned by the analysis functions. The projection is equivalent to a weighted least squares problem, where the weights are induced from the certainty of the signal and the desired localization of the analysis functions. The result of normalized convolution is at each signal point a set of expansion coefficients, one for each analysis function.

While neither least squares fitting, localization of analysis functions, nor handling of uncertain data in themselves are novel ideas, the unique strength of normalized convolution is that it combines all of them simultaneously in a well structured and theoretically sound way. The method is a generally useful tool for signal analysis in the spatial domain, which formalizes and generalizes least squares techniques, e.g. the facet model [41, 42, 43, 44], that have been used for a long time.

A weakness of the presentation used here is that it is not immediately clear how

normalized convolution can be applied to the problem of computing filter responses on uncertain data when the convolution kernels are given. This is discussed in detail in section 3.8.

3.2 Definition of Normalized Convolution

Before defining normalized convolution, it is necessary to get familiar with the terms signal, certainty, basis functions, and applicability, in the context of the method. To begin with we assume that we have discrete signals, and explore the straightforward generalization to continuous signals in section 3.2.4.

3.2.1 Signal and Certainty

It is important to be aware that normalized convolution can be considered as a pointwise operation, or more strictly, as an operation on a neighborhood of each signal point. This is no different from ordinary convolution, where the convolution result at each point is effectively the inner product between the conjugated and reflected filter kernel and a neighborhood of the signal.

Let f denote the whole signal while $\mathbf{f}$ denotes the neighborhood of a given point. It is assumed that the neighborhood is of finite size[1], so that $\mathbf{f}$ can be considered an element of a finite dimensional vector space $\mathcal{C}^n$. Regardless of the dimensionality of the space in which it is embedded[2], $\mathbf{f}$ is represented by an $n \times 1$ column vector.[3]

Certainty is a measure of the confidence in the signal values at each point, given by non-negative real numbers. Let c denote the whole certainty field, while the $n \times 1$ column vector $\mathbf{c}$ denotes the certainty of the signal values in $\mathbf{f}$.

Possible causes for uncertainty in signal values are, e.g., defective sensor elements, detected (but not corrected) transmission errors, and varying confidence in the results from previous processing. The most important, and rather ubiquitous case of uncertainty, however, is missing data outside the border of the signal, so called border effects. The problem is that for a signal of limited extent, the neighborhood of points close to the border will include points where no values are given. This has traditionally been handled in a number of different ways. The most common is to assume that the signal values are zero outside the border, which implicitly is done by standard convolution. Another way is to assume cyclical repetition of the signal values, which implicitly is done when convolution is computed in the frequency domain. Yet another way is to extend with the values at the border. None of these is completely satisfactory, however. The correct way to do it, from a signal/certainty perspective, is to set the certainty for points outside the border to zero, while the signal value is left unspecified.

[1] This condition can be lifted, as discussed in section 3.2.4. For computational reasons, however, it is in practice always satisfied.

[2] E.g. dimensionality 2 for image data.

[3] The elements of the vector are implicitly the coordinates relative to some orthonormal basis, typically a pixel basis.

It is obvious that certainty zero means missing data, but it is not so clear how positive values should be interpreted. An exact interpretation must be postponed until section 3.2.4, but of course a larger certainty corresponds to a higher confidence in the signal value. It may seem natural to limit certainty values to the range $[0, 1]$, with 1 meaning full confidence, but this is not really necessary.

3.2.2 Basis Functions and Applicability

The role of the basis functions is to give a local model for the signal. Each basis function has the size of the neighborhood mentioned above, i.e. it is an element of $\mathcal{C}^n$, represented by an $n \times 1$ column vector $\mathbf{b}_i$. The set $\{\mathbf{b}_i\}_1^m$ of basis functions are stored in an $n \times m$ matrix $\mathbf{B}$,

$$\mathbf{B} = \begin{pmatrix} | & | & & | \\ \mathbf{b}_1 & \mathbf{b}_2 & \dots & \mathbf{b}_m \\ | & | & & | \end{pmatrix}. \tag{3.1}$$

Usually we have linearly independent basis functions, so that the vectors $\{\mathbf{b}_i\}$ do constitute a basis for a subspace of $\mathcal{C}^n$. In most cases m is also much smaller than n.

The applicability is a kind of "certainty" for the basis functions. Rather than being a measure of certainty or confidence, however, it indicates the significance or importance of each point in the neighborhood. Like the certainty, the applicability $\mathbf{a}$ is represented by an $n \times 1$ vector with non-negative elements. Points where the applicability is zero might as well be excluded from the neighborhood altogether, but for practical reasons it may be convenient to keep them. As for certainty it may seem natural to limit the applicability values to the range $[0, 1]$ but there is really no reason to do this because the scaling of the values turns out to be of no significance.

The basis functions may actually be defined for a larger domain than the neighborhood in question. They can in fact be unlimited, e.g. polynomials or complex exponentials, but values at points where the applicability is zero simply do not matter. This is an important role of the applicability, to enforce a spatial localization of the signal model. A more extensive discussion on the choice of applicability follows in section 3.10.

3.2.3 Definition

Let the $n \times n$ matrices $\mathbf{W}_a = \text{diag}\,(\mathbf{a})$, $\mathbf{W}_c = \text{diag}\,(\mathbf{c})$, and $\mathbf{W}^2 = \mathbf{W}_a\mathbf{W}_c$.[4] *The operation of normalized convolution is at each signal point a question of representing a neighborhood of the signal,* $\mathbf{f}$*, by the set of vectors* $\{\mathbf{b}_i\}$*, using the weighted norm (or seminorm)* $\|\cdot\|_{\mathbf{W}}$ *in the signal space and the Euclidian norm in the coefficient space.* The result of normalized convolution is at each point the set of coefficients $\mathbf{r}$ used in the vector set representation of the neighborhood.

[4] We set $\mathbf{W}^2 = \mathbf{W}_a\mathbf{W}_c$ to keep in line with the established notation. Letting $\mathbf{W} = \mathbf{W}_a\mathbf{W}_c$ would be equally valid, as long as $\mathbf{a}$ and $\mathbf{c}$ are interpreted accordingly, and somewhat more natural in the framework used here.

As we have seen in chapter 2, this can equivalently be stated as the seminorm weighted general linear least squares problem

$$\arg\min_{\mathbf{r}\in\mathcal{S}} \|\mathbf{r}\|, \qquad \mathcal{S} = \{\mathbf{r}\in\mathcal{C}^m; \|\mathbf{Br}-\mathbf{f}\|_{\mathbf{W}} \text{ is minimum}\}. \tag{3.2}$$

In the case that the basis functions are linearly independent with respect to the (semi)norm $\|\cdot\|_{\mathbf{W}}$, this can be simplified to the more ordinary weighted linear least squares problem

$$\arg\min_{\mathbf{r}\in\mathcal{C}^m} \|\mathbf{Br}-\mathbf{f}\|_{\mathbf{W}}. \tag{3.3}$$

In any case the solution is given by equation (2.64) as

$$\mathbf{r} = (\mathbf{WB})^{\dagger}\mathbf{Wf}. \tag{3.4}$$

For various purposes it is useful to rewrite this formula. We start by expanding the pseudo-inverse in (3.4) by identity (2.16), leading to

$$\mathbf{r} = (\mathbf{B}^*\mathbf{W}^2\mathbf{B})^{\dagger}\mathbf{B}^*\mathbf{W}^2\mathbf{f}, \tag{3.5}$$

which can be interpreted in terms of inner products as

$$\mathbf{r} = \begin{pmatrix} (\mathbf{b}_1,\mathbf{b}_1)_{\mathbf{W}} & \dots & (\mathbf{b}_1,\mathbf{b}_m)_{\mathbf{W}} \\ \vdots & \ddots & \vdots \\ (\mathbf{b}_m,\mathbf{b}_1)_{\mathbf{W}} & \dots & (\mathbf{b}_m,\mathbf{b}_m)_{\mathbf{W}} \end{pmatrix}^{\dagger} \begin{pmatrix} (\mathbf{b}_1,\mathbf{f})_{\mathbf{W}} \\ \vdots \\ (\mathbf{b}_m,\mathbf{f})_{\mathbf{W}} \end{pmatrix}. \tag{3.6}$$

Replacing $\mathbf{W}^2$ with $\mathbf{W}_a\mathbf{W}_c$ and using the assumption that the vectors $\{\mathbf{b}_i\}$ constitute a subspace basis with respect to the (semi)norm $\mathbf{W}$, so that the pseudo-inverse in (3.5) and (3.6) can be replaced with an ordinary inverse, we get

$$\mathbf{r} = (\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f} \tag{3.7}$$

and with the convention that $\cdot$ denotes pointwise multiplication, we arrive at the expression[5]

$$\mathbf{r} = \begin{pmatrix} (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1,\mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1,\mathbf{b}_m) \\ \vdots & \ddots & \vdots \\ (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m,\mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m,\mathbf{b}_m) \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1,\mathbf{f}) \\ \vdots \\ (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m,\mathbf{f}) \end{pmatrix}. \tag{3.8}$$

3.2.4 Comments on the Definition

In previous formulations of normalized convolution, it has been assumed that the basis functions do constitute a subspace basis, so that we have a unique solution to the linear least squares problem (3.3), given by (3.7) or (3.8). The problem with this assumption is that if we have a neighborhood with lots of missing data,

[5] This is almost the original formulation of normalized convolution. The final step is postponed until section 3.3.

it can happen that the basis functions effectively become linearly dependent in the seminorm given by $\mathbf{W}$, so that the inverses in (3.7) and (3.8) do not exist.

We can solve this problem by exchanging the inverses for pseudo-inverses, equations (3.5) and (3.6), which removes the ambiguity in the choice of resulting coefficients $\mathbf{r}$ by giving the solution to the more general linear least squares problem (3.2). This remedy is not without risks, however, since the mere fact that the basis functions turn linearly dependent, indicates that the values of at least some of the coefficients may be very uncertain. More discussion on this follows in section 3.5. Taking proper care in the interpretation of the result, however, the pseudo-inverse solutions should be useful when the signal certainty is very low. They are also necessary in certain generalizations of normalized convolution, see section 3.12.

To be able to use the framework from chapter 2 in deriving the expressions for normalized convolution, we restricted ourselves to the case of discrete signals and neighborhoods of finite size. When we have continuous signals and/or infinite neighborhoods we can still use (3.6) or (3.8) to define normalized convolution, simply by using an appropriate weighted inner product. The corresponding least squares problems are given by obvious modifications to (3.2) and (3.3).

The geometrical interpretation of the least squares minimization is that the local neighborhood is projected into the subspace spanned by the basis functions, using a metric that is dependent on the certainty and the applicability. From the least squares formulation we can also get an exact interpretation of the certainty and the applicability. The certainty gives the relative importance of the signal values when doing the least squares fit, while the applicability gives the relative importance of the points in the neighborhood. Obviously a scaling of the certainty or applicability values does not change the least squares solution, so there is no reason to restrict these values to the range $[0, 1]$.

3.3 Implementational Issues

While any of the expressions (3.4)–(3.8) can be used to compute normalized convolution, there are some differences with respect to computational complexity and numeric stability. Numerically (3.4) is somewhat preferable to the other expressions, because values get squared in the rest of them, raising the condition numbers. Computationally, however, the computation of the pseudo-inverse is costly and $\mathbf{WB}$ is typically significantly larger than $\mathbf{B}^*\mathbf{W}^2\mathbf{B}$. We rather wish to avoid the pseudo-inverses altogether, leaving us with (3.7) and (3.8). The inverses in these expressions are of course not computed explicitly, since there are more efficient methods to solve linear equation systems. In fact, the costly operation now is to compute the inner products in (3.8). Remembering that these computations have to be performed at each signal point, we can improve the expression somewhat by rewriting (3.8) as

$$\mathbf{r} = \begin{pmatrix} (\mathbf{a} \cdot \mathbf{b}_1 \cdot \bar{\mathbf{b}}_1, \mathbf{c}) & \dots & (\mathbf{a} \cdot \mathbf{b}_1 \cdot \bar{\mathbf{b}}_m, \mathbf{c}) \\ \vdots & \ddots & \vdots \\ (\mathbf{a} \cdot \mathbf{b}_m \cdot \bar{\mathbf{b}}_1, \mathbf{c}) & \dots & (\mathbf{a} \cdot \mathbf{b}_m \cdot \bar{\mathbf{b}}_m, \mathbf{c}) \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{a} \cdot \mathbf{b}_1, \mathbf{c} \cdot \mathbf{f}) \\ \vdots \\ (\mathbf{a} \cdot \mathbf{b}_m, \mathbf{c} \cdot \mathbf{f}) \end{pmatrix}, \tag{3.9}$$

where $\bar{\mathbf{b}}_i$ denotes complex conjugation of the basis functions. This is actually the original formulation of normalized convolution [64, 66, 94], although with different notation. By precomputing the quantities $\{\mathbf{a} \cdot \mathbf{b}_k \cdot \bar{\mathbf{b}}_l\}$, $\{\mathbf{a} \cdot \mathbf{b}_k\}$, and $\mathbf{c} \cdot \mathbf{f}$, we can decrease the total number of multiplications at the cost of a small increase in storage requirements.

To compute normalized convolution at all points of the signal we essentially have two strategies. The first is to compute all inner products and to solve the linear equation system for one point before continuing to the next point. The second is to compute the inner product for all points before continuing with the next inner product and at the very last solve all the linear equation systems. The advantage of the latter approach is that the inner products can be computed as standard convolutions, an operation which is often available in heavily optimized form, possibly in hardware. The disadvantage is that large amounts of extra storage must be used, which even if it is available could lead to problems with respect to data locality and cache performance. Further discussion on how normalized convolution can be computed significantly more efficiently in certain cases can be found in sections 3.7, 4.3, and 4.4.

3.4 Example

To give a small example, assume that we have a two-dimensional signal f, sampled at integer points, with an accompanying certainty field c, as defined below.

$$f = \begin{matrix} 3 & 7 & 4 & 5 & 8 \\ 9 & 2 & 4 & 4 & 6 \\ 5 & 1 & 4 & 3 & 7 \\ 3 & 1 & 1 & 2 & 8 \\ 4 & 6 & \boxed{2} & 3 & 6 \\ 7 & 3 & 2 & 6 & 3 \\ 9 & 6 & 4 & 9 & 9 \end{matrix} \qquad c = \begin{matrix} 0 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 2 & 2 \\ 2 & 1 & 1 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 0 & \boxed{2} & 2 & 2 \\ 1 & 1 & 2 & 1 & 0 \\ 2 & 2 & 2 & 1 & 0 \end{matrix} \tag{3.10}$$

Let the local signal model be given by a polynomial basis, $\{1, x, y, x^2, xy, y^2\}$ (it is understood that the x variable increases from the left to the right, while the y variable increases going downwards) and an applicability of the form:

$$a = \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \tag{3.11}$$

The applicability fixes the size of the neighborhood, in this case 3×3 pixels, and gives a localization of the unlimited polynomial basis functions. Expressed as

matrices, taking the points columnwise, we have

$$\mathbf{B} = \begin{pmatrix} 1 & -1 & -1 & 1 & 1 & 1 \\ 1 & -1 & 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 1 & -1 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \text{ and } \mathbf{a} = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 4 \\ 2 \\ 1 \\ 2 \\ 1 \end{pmatrix}. \tag{3.12}$$

Assume that we wish to compute the result of normalized convolution at the marked point in the signal. Then the signal and certainty neighborhoods are represented by

$$\mathbf{f} = \begin{pmatrix} 1 \\ 6 \\ 3 \\ 1 \\ 2 \\ 2 \\ 2 \\ 3 \\ 6 \end{pmatrix} \text{ and } \mathbf{c} = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}. \tag{3.13}$$

Applying equation (3.7) we get the result

$$\begin{aligned} \mathbf{r} &= (\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f} \\ &= \begin{pmatrix} 26 & 4 & -2 & 10 & 0 & 14 \\ 4 & 10 & 0 & 4 & -2 & 0 \\ -2 & 0 & 14 & -2 & 0 & -2 \\ 10 & 4 & -2 & 10 & 0 & 6 \\ 0 & -2 & 0 & 0 & 6 & 0 \\ 14 & 0 & -2 & 6 & 0 & 14 \end{pmatrix}^{-1} \begin{pmatrix} 55 \\ 17 \\ 7 \\ 27 \\ 1 \\ 27 \end{pmatrix} = \begin{pmatrix} 1.81 \\ 0.72 \\ 0.86 \\ 0.85 \\ 0.41 \\ -0.12 \end{pmatrix} \end{aligned} \tag{3.14}$$

As we will see in chapter 5, with this choice of basis functions, the resulting coefficients hold much information on the the local orientation of the neighborhood. To conclude this example, we reconstruct the projection of the signal, $\mathbf{Br}$, and reshape it to a 3×3 neighborhood:

$$\begin{matrix} 1.36 & 0.83 & 1.99 \\ 1.94 & 1.81 & 3.38 \\ 2.28 & 2.55 & 4.52 \end{matrix} \tag{3.15}$$

To get the result of normalized convolution at all points of the signal, we repeat the above process at each point.

3.5 Output Certainty

To be consistent with the signal/certainty philosophy, the result of normalized convolution should of course be accompanied by an output certainty. Unfortunately, this is for the most part an open problem.

Factors that ought to influence the output certainty at a given point include

1. the amount of input certainty in the neighborhood,
2. the sensitivity of the result to noise, and
3. to which extent the signal can be described by the basis functions.

The sensitivity to noise is smallest when the basis functions are orthogonal[6], because the resulting coefficients are essentially independent. Should two basis functions be almost parallel, on the other hand, they both tend to get relatively large coefficients, and input noise in a certain direction gets amplified.

Two possible measures of output certainty have been published, by Westelius [93] and Karlholm [61] respectively. Westelius has used

$$c_{\text{out}} = \left(\frac{\det \mathbf{G}}{\det \mathbf{G}_0} \right)^{\frac{1}{m}}, \tag{3.16}$$

while Karlholm has used

$$c_{\text{out}} = \frac{1}{\|\mathbf{G}_0\|_2 \|\mathbf{G}^{-1}\|_2}. \tag{3.17}$$

In both expressions we have

$$\mathbf{G} = \mathbf{B}^* \mathbf{W}_a \mathbf{W}_c \mathbf{B} \quad \text{and} \quad \mathbf{G}_0 = \mathbf{B}^* \mathbf{W}_a \mathbf{B}, \tag{3.18}$$

where $\mathbf{G}_0$ is the same as $\mathbf{G}$ if the certainty is identically one.

Both these measures take the points 1 and 2 above into account. A disadvantage, however, is that they give a single certainty value for all the resulting coefficients, which makes sense with respect to 1 but not with respect to the sensitivity issues. Clearly, if we have two basis functions that are nearly parallel, but the rest of them are orthogonal, we have good reason to mistrust the coefficients corresponding to the two almost parallel basis functions, but not necessarily the rest of the coefficients.

A natural measure of how well the signal can be described by the basis functions is given by the residual error in (3.2) or (3.3),

$$\|\mathbf{B}\mathbf{r} - \mathbf{f}\|_{\mathbf{W}}. \tag{3.19}$$

In order to be used as a measure of output certainty, some normalization with respect to the amplitude of the signal and the input certainty should be performed.

One thing to be aware of in the search for a good measure of output certainty, is that it probably must depend on the application, or more precisely, on how the result is further processed.

[6] Remember that orthogonality depends on the inner product, which in turn depends on the certainty and the applicability.

3.6 Normalized Differential Convolution

When doing signal analysis, it may be important to be invariant to certain irrelevant features. A typical example can be seen in chapter 5, where we want to estimate the local orientation of a multidimensional signal. It is clear that the local DC level gives no information about the orientation, but we cannot simply ignore it because it would affect the computation of the interesting features. The solution is to include the features to which we wish to be invariant in the signal model. This means that we expand the set of basis functions, but ignore the corresponding coefficients in the result.

Since we do not care about some of the resulting coefficients, it may seem wasteful to use (3.7), which computes all of them. To avoid this we start by partitioning

$$\mathbf{B} = \begin{pmatrix} \mathbf{B}_1 & \mathbf{B}_2 \end{pmatrix} \quad \text{and} \quad \mathbf{r} = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{pmatrix}, \tag{3.20}$$

where $\mathbf{B}_1$ contains the basis functions we are interested in, $\mathbf{B}_2$ contains the basis functions to which we wish to be invariant, and $\mathbf{r}_1$ and $\mathbf{r}_2$ are the corresponding parts of the resulting coefficients. Now we can rewrite (3.7) in partitioned form as

$$\begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{B}_1^* \mathbf{W}_a \mathbf{W}_c \mathbf{B}_1 & \mathbf{B}_1^* \mathbf{W}_a \mathbf{W}_c \mathbf{B}_2 \\ \mathbf{B}_2^* \mathbf{W}_a \mathbf{W}_c \mathbf{B}_1 & \mathbf{B}_2^* \mathbf{W}_a \mathbf{W}_c \mathbf{B}_2 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{B}_1^* \mathbf{W}_a \mathbf{W}_c \mathbf{f} \\ \mathbf{B}_2^* \mathbf{W}_a \mathbf{W}_c \mathbf{f} \end{pmatrix} \tag{3.21}$$

and continue the expansion with an explicit expression for the inverse of a partitioned matrix [60],

$$\begin{gathered} \begin{pmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^* & \mathbf{B} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} + \mathbf{E}\boldsymbol{\Delta}^{-1}\mathbf{F} & -\mathbf{E}\boldsymbol{\Delta}^{-1} \\ -\boldsymbol{\Delta}^{-1}\mathbf{F} & \boldsymbol{\Delta}^{-1} \end{pmatrix}, \\ \boldsymbol{\Delta} = \mathbf{B} - \mathbf{C}^*\mathbf{A}^{-1}\mathbf{C}, \quad \mathbf{E} = \mathbf{A}^{-1}\mathbf{C}, \quad \mathbf{F} = \mathbf{C}^*\mathbf{A}^{-1}. \end{gathered} \tag{3.22}$$

The resulting algorithm is called normalized differential convolution [61, 64, 66, 94, 95]. The primary advantage over the expression for normalized convolution is that we get smaller matrices to invert, but on the other hand we need to actually compute the inverses here[7], instead of just solving a single linear equation system, and there are also more matrix multiplications to perform. It seems unlikely that it would be worth the extra complications to avoid computing the uninteresting coefficients, unless $\mathbf{B}_1$ and $\mathbf{B}_2$ contain only a single vector each, in which case the expression for normalized differential convolution simplifies considerably.

In the following chapters we use the basic normalized convolution, even if we are not interested in all the coefficients.

[7] This is not quite true, since it is sufficient to compute factorizations that allow us to solve corresponding linear equation systems, but we need to solve several of these instead of just one.

3.7 Reduction to Ordinary Convolution

If we have the situation that the certainty field remains fixed while the signal varies, we can save a lot of work by precomputing the matrices

$$\tilde{\mathbf{B}}^* = (\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c \tag{3.23}$$

at every point, at least if we can afford the extra storage necessary. A possible scenario for this situation is that we have a sensor array where we know that certain sensor elements are not working or give less reliable measurements. Another case, which is very common, is that we simply do not have any certainty information at all and can do no better than setting the certainty for all values to one. Notice, however, that if the extent of the signal is limited, we have certainty zero outside the border. In this case we have the same certainty vector for many neighborhoods and only have to compute and store a small number of different $\tilde{\mathbf{B}}$.

As can be suspected from the notation, $\tilde{\mathbf{B}}$ can be interpreted as a dual basis matrix. Unfortunately it is not the weighted dual subspace basis given by (2.53), because the resulting coefficients are computed by $(\tilde{\mathbf{b}}_i, \mathbf{f})$ rather than by using the proper[8] inner product $(\tilde{\mathbf{b}}_i, \mathbf{f})_{\mathbf{W}}$. We will still use the term dual vectors here, although somewhat improperly.

If we assume that we have constant certainty one and restrict ourselves to compute normalized convolution for the part of the signal that is sufficiently far from the border, we can reduce normalized convolution to ordinary convolution. At each point the result can be computed as

$$\mathbf{r} = \tilde{\mathbf{B}}^*\mathbf{f} \tag{3.24}$$

or coefficient by coefficient as

$$r_i = (\tilde{\mathbf{b}}_i, \mathbf{f}). \tag{3.25}$$

Extending these computations over all points under consideration, we can write

$$r_i(\mathbf{x}) = (\tilde{\mathbf{b}}_i, \mathcal{T}_{\mathbf{x}} f), \tag{3.26}$$

where $\mathcal{T}_{\mathbf{x}}$ is a translation operator, $\mathcal{T}_{\mathbf{x}} f(\mathbf{u}) = f(\mathbf{u} + \mathbf{x})$. This expression can in turn be rewritten as a convolution

$$r_i(\mathbf{x}) = (\check{\tilde{\mathbf{b}}}_i * f)(\mathbf{x}), \tag{3.27}$$

where we let $\check{\tilde{\mathbf{b}}}_i$ denote the conjugated and reflected $\tilde{\mathbf{b}}_i$.

The need to reflect and conjugate the dual basis functions in order to get convolution kernels is a complication that we would prefer to avoid. We can do this by replacing the convolution with an unnormalized cross correlation, using the notation from Bracewell [15],

$$(g \star h)(\mathbf{x}) = \sum_{\mathbf{u}} \bar{g}(\mathbf{u}) h(\mathbf{u} + \mathbf{x}). \tag{3.28}$$

[8] Proper with respect to the norm used in the minimization (3.3).

With this operation, (3.27) can be rewritten as

$$r_i(\mathbf{x}) = (\tilde{\mathbf{b}}_i \star f)(\mathbf{x}). \tag{3.29}$$

The cross correlation is in fact a more natural operation to use in this context than ordinary convolution, since we are not much interested in the properties that otherwise give convolution an advantage. We have, e.g., no use for the property that $g * h = h * g$, since we have a marked asymmetry between the signal and the basis functions. The ordinary convolution is, however, a much more well known operation, so while we will use the cross correlation further on, it is useful to remember that we get the corresponding convolution kernels simply by conjugating and reflecting the dual basis functions.

To get a better understanding of the dual basis functions, we can rewrite (3.23), with $\mathbf{W}_c = \mathbf{I}$, as

$$\begin{pmatrix} | & | & & | \\ \tilde{\mathbf{b}}_1 & \tilde{\mathbf{b}}_2 & \dots & \tilde{\mathbf{b}}_m \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{a}\cdot\mathbf{b}_1 & \mathbf{a}\cdot\mathbf{b}_2 & \dots & \mathbf{a}\cdot\mathbf{b}_m \\ | & | & & | \end{pmatrix} \mathbf{G}_0^{-1}, \tag{3.30}$$

where $\mathbf{G}_0 = \mathbf{B}^*\mathbf{W}_a\mathbf{B}$ as in (3.18). Hence we obtain the duals as linear combinations of the basis functions $\mathbf{b}_i$, windowed by the applicability $\mathbf{a}$. The role of $\mathbf{G}_0^{-1}$ is to compensate for dependencies between the basis functions when they are not orthogonal. Notice that this includes non-orthogonalities caused by the windowing by $\mathbf{a}$. A concrete example of dual basis functions can be found in section 4.3.1.

3.8 Filter Responses on Uncertain Data

The previous section showed how normalized convolution can be reduced to ordinary convolution. The converse problem is also important. I.e. given a set of convolution kernels, how can we use normalized convolution to compute filter responses on uncertain data?

3.8.1 Corresponding Basis Functions

Assume that we have a set of convolution kernels given, which we conjugate and reflect to obtain the equivalent correlation kernels $\{\mathbf{h}_i\}_1^m$, so that

$$r_i(\mathbf{x}) = (\mathbf{h}_i \star f)(\mathbf{x}). \tag{3.31}$$

The idea is now to find a set of basis functions and an applicability with the property that in the case of full certainty, the dual basis functions coincide with the given correlation kernels. Thus normalized convolution using the computed basis functions and applicability gives the expected results when all data are certain and something that is expected to be useful when uncertain data are present.

To solve this problem we first collect the correlation kernels $\{\mathbf{h}_i\}$ into the matrix $\mathbf{H}$. Since we want this to coincide with $\tilde{\mathbf{B}}$ in the previous section, (3.23) with $\mathbf{W}_c = \mathbf{I}$ gives the equation

$$\mathbf{H} = \mathbf{W}_a\mathbf{B}(\mathbf{B}^*\mathbf{W}_a\mathbf{B})^{-1}, \tag{3.32}$$

which we wish to solve for $\mathbf{B}$ and $\mathbf{W}_a$. Somewhat surprisingly we can find a $\mathbf{B}$ satisfying this equation for any invertible $\mathbf{W}_a$, with the explicit and easily verified solution

$$\mathbf{B} = \mathbf{W}_a^{-1}\mathbf{H}(\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{H})^{-1}. \tag{3.33}$$

An additional requirement is that the correlation kernels are linearly independent, so that $\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{H}$ is invertible. Now that we have this $\mathbf{B}$, normalized convolution is computed as usual and the expansion coefficients give the filter responses. Obviously we must use the same applicability as in the construction of $\mathbf{B}$.

3.8.2 Practical Considerations

While the presented procedure is simple and straightforward, it must be used with care. There are several pitfalls, some of them not at all obvious.

- One observation is that we can choose the applicability arbitrarily as long as all values are non-zero. As is discussed in section 3.10, very little can be said in general about this choice. One thing to be aware of in this situation though, is that the presence of very small applicability values invites numerical problems in (3.33).

- An important property of certain filters is that they have a DC response which is exactly or very close to zero. This is e.g. the case for gradient filters. It is easy to show that this property in general will be lost when filter responses are computed in the proposed way on signals with varying certainty. There is, however, a simple and important workaround for this problem. The solution is to add one extra basis function which is identically one. Clearly any DC variation in the signal will project solely onto this basis function so the rest of the expansion coefficients (i.e. the filter responses) will then have zero DC response. This is incidentally the classic application of normalized differential convolution, section 3.6. Conceivably one may also want to guarantee invariance to more features than DC variations.

- Assuming we have a set of m filters, the proposed procedure gives m corresponding basis functions which must be used together when computing the normalized convolution. A plausible alternative would be to apply the construction of corresponding basis functions for one convolution kernel at a time and compute the filter responses through m normalized convolutions with a single basis function.

  Considering that the number of inner products in equation (3.9) grows with the square of the number of basis functions, it is easy to see that the latter approach has computational advantages. Before entering this path, however, one should be aware that these two approaches in general give different results. It is not obvious exactly what the significance of these differences are, but experience seems to indicate that if the filters in the filter set are designed to work together, it is probably more correct not to split them up.

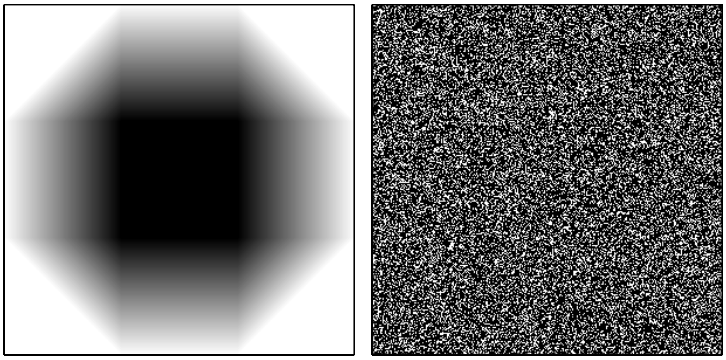
- Another interesting question is how to deal with complex filters. For this discussion it suffices to assume that we have just a single complex filter, with a correlation kernel $\mathbf{h}_c = \mathbf{h}_r + i\mathbf{h}_i$. Equation (3.33) is valid also for complex correlation kernels, so this does not lead to any complications and we obtain a complex corresponding basis function $\mathbf{b}_c$. We do, however, also have the option to consider the complex filter as a set of two real filters, $\{\mathbf{h}_r, \mathbf{h}_i\}$. Once more applying equation (3.33), we obtain the two corresponding basis functions $\mathbf{b}_r$ and $\mathbf{b}_i$. It is easy to verify that in general $\mathbf{b}_c$ will differ from $\mathbf{b}_r + i\mathbf{b}_i$ and more importantly that the filter responses after normalized convolution will be different. We can also see that the first approach is less computationally expensive, so we have a situation similar to the previous one. Here too, it is not obvious what significance the differences have, but we do believe that the latter, more expensive approach, usually is the more correct one. However, the quality of the results must always be weighed against the computational cost. What choice to make depends very much on the application. This trade-off is also discussed by Westelius in [93] in the context of quadrature filters. His conclusion is that while real filtering gives better results, it is not worth the increased computational complexity in his application.

To illustrate these points, we use two simple orthogonal gradient filters, with correlation kernels

$$\mathbf{h}_x(x, y) = xg(x, y), \tag{3.34}$$
$$\mathbf{h}_y(x, y) = yg(x, y), \tag{3.35}$$

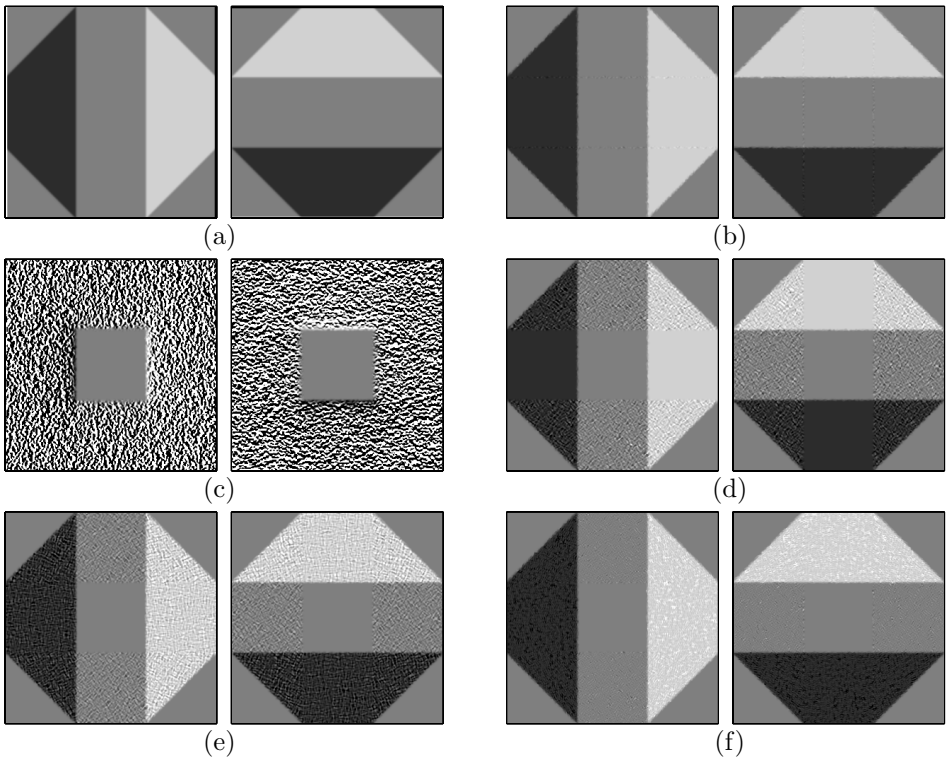
where $g(x, y)$ is a 9×9 Gaussian with standard deviation 1.2. Figure 3.1(a) shows a simple test image with linear ramps in different directions and flat areas in the center and in the corners. To simulate missing data we use the binary certainty field shown in figure 3.1(b), which has an average density of 30%.

Figure 3.2 shows the filtering results. In (a) we have the original filter responses, without missing data. Notice the artefacts along the borders. In (b) we have the result of applying the proposed algorithm with both kernels together, adding a constant basis function, and using a Gaussian applicability identical to $g(x, y)$. This result is even better than the original filter responses because the missing data off the borders is correctly taken into account. The disastrous effect of neglecting to add a constant basis function is shown in (c). The results are so bad that most of the response image is saturated. If we add back the constant basis function and instead compute the filter responses separately we get the result in (d). This is similar to the result of doing the computations for a single complex filter $\mathbf{h}_x + i\mathbf{h}_y$, shown in (e). Finally, (f) shows that the choice of applicability in (b) was ideal. Here a Gaussian applicability with standard deviation 1.4 is used instead. The explanation for the exceptional result in (b) is that it exactly corresponds to normalized convolution with the basis functions $\{1, x, y\}$ on a signal which perfectly follows the signal model.



(a) (b)

Figure 3.1: Test image (a) and certainty field (b).



(a) (b) (c) (d) (e) (f)

Figure 3.2: Filtering results on uncertain data.

3.8.3 Alternative Interpretations

Inserting equation (3.33) into (3.7) gives an expression for the filter response at one point directly in the correlation kernel matrix $\mathbf{H}$,

$$\mathbf{r} = \mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{H}(\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{W}_c\mathbf{H})^{-1}\mathbf{H}^*\mathbf{W}_c\mathbf{f}. \tag{3.36}$$

Using basis functions obtained by dividing the correlation kernels pointwise with the applicability, i.e. $\mathbf{B} = \mathbf{W}_a^{-1}\mathbf{H}$, we get

$$\mathbf{r} = \mathbf{B}^*\mathbf{W}_a\mathbf{B}(\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f} = \mathbf{G}_0(\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f}, \tag{3.37}$$

which can be interpreted as the usual result of normalized convolution transformed by the matrix $\mathbf{G}_0$ from equation (3.18). This is the approach used in [93].

Equation (3.37) can also be rewritten

$$\mathbf{r} = \mathbf{H}^*(\mathbf{B}(\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f}) = \mathbf{H}^*\mathbf{f}_{\text{proj}}, \tag{3.38}$$

with the interpretation that $\mathbf{f}$ is projected onto the basis $\mathbf{B} = \mathbf{W}_a^{-1}\mathbf{H}$ using normalized convolution to get $\mathbf{f}_{\text{proj}}$, which is then filtered by standard correlation (or convolution) to obtain the filter responses $\mathbf{r}$.

3.9 Application Examples

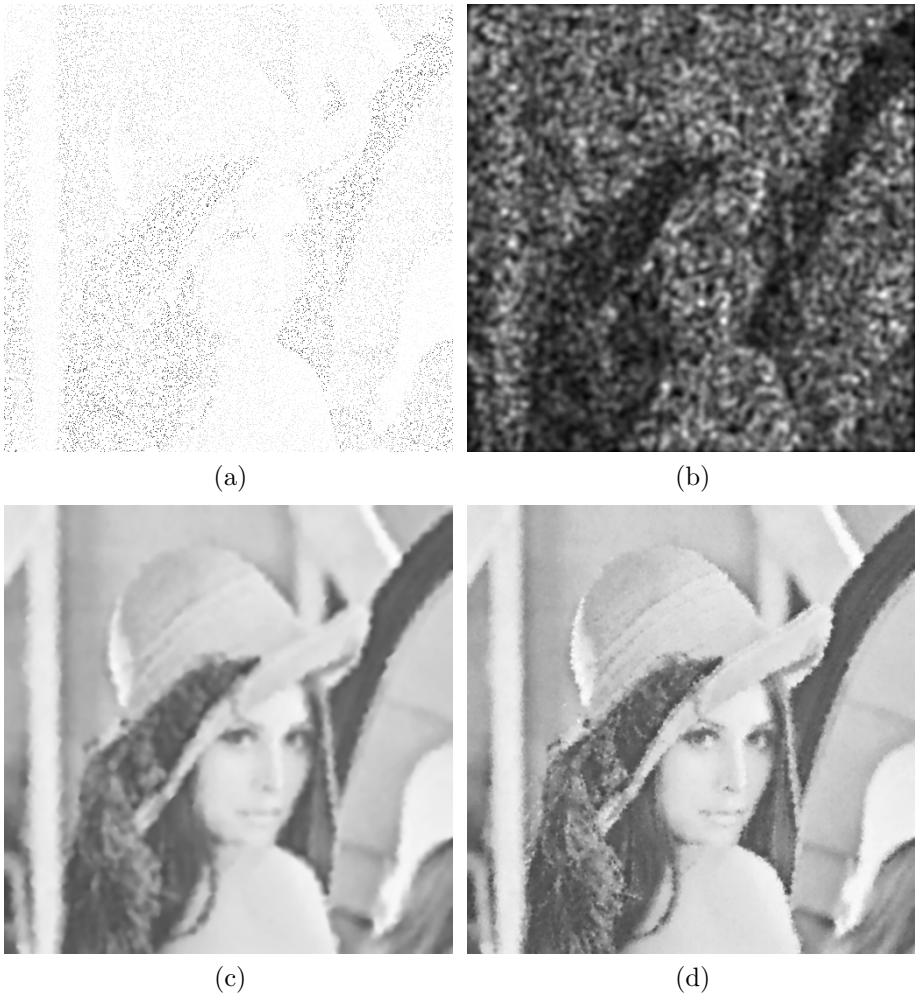
Applications where normalized convolution has been used include interpolation [66, 94, 98], frequency estimation [65], estimation of texture gradients [84], gradient estimation [58], edge detection [37], increased selectivity for rotational symmetry detection [54, 57], depth segmentation [85, 86], motion compensated prediction [3, 4], phase-based stereopsis and focus-of-attention control [93, 96], and orientation and motion estimation [61]. In the two latter applications, normalized convolution is utilized to compute quadrature filter responses on uncertain data.

3.9.1 Normalized Averaging

The most striking example is perhaps, despite its simplicity, normalized averaging to interpolate missing samples in an image. We illustrate this technique with a partial reconstruction of the example given in [40, 66, 94, 98].

In figure 3.3(a) the well-known Lena image has been degraded so that only 10% of the pixels remain.[9] The remaining pixels have been selected randomly with uniform distribution from a 512×512 grayscale original. Standard convolution with a smoothing filter, given by figure 3.4(a), leads to a highly non-satisfactory result, figure 3.3(b), because no compensation is made for the variation in local sampling density. An ad hoc solution to this problem would be to divide the previous convolution result with the convolution between the smoothing filter and the certainty field, with the latter being an estimate of the local sampling density.

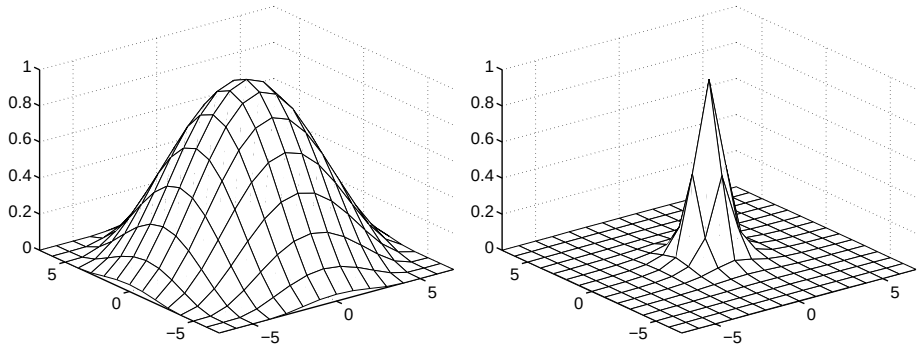
[9] The removed pixels have been replaced by zeros, i.e. black. For illustration purposes, missing samples are rendered white in figures 3.3(a) and 3.5(a).



(a) (b)

(c) (d)

Figure 3.3: Normalized averaging. (a) Degraded test image, only 10% of the pixels remain. (b) Standard convolution with smoothing filter. (c) Normalized averaging with applicability given by figure 3.4(a). (d) Normalized averaging with applicability given by figure 3.4(b).



(a) $\mathbf{a} = \begin{cases} \cos^2 \frac{\pi r}{16}, & r < 8 \\ 0, & \text{otherwise} \end{cases}$ (b) $\mathbf{a} = \begin{cases} 1, & r < 1 \\ 0.5 r^{-3}, & 1 \leq r < 8 \\ 0, & \text{otherwise} \end{cases}$

Figure 3.4: Applicability functions used for normalized averaging.

(a) (b)

Figure 3.5: Normalized averaging on an inhomogeneously sampled image. (a) Degraded test image, only 4% of the pixels remain. (b) Normalized averaging with applicability given by figure 3.4(b).

This idea can easily be formalized by means of normalized convolution. The signal and the certainty are already given. We use a single basis function, a constant one, and use the smoothing filter as the applicability.[10] The result from this operation, figure 3.3(c), can be interpreted as a weighted and normalized average of the pixels present in the neighborhood, and is identical to the ad hoc solution above. In figure 3.3(d) we see the result of normalized averaging with a more localized applicability, given by figure 3.4(b).

To expand on the example, we notice that instead of having a uniform distribution of the remaining pixels, it would be advantageous to have more samples in areas of high contrast. Figure 3.5(a) shows such a test image, only containing 4% of the original pixels. The result of normalized averaging, with applicability given by figure 3.4(b), is shown in figure 3.5(b).

3.9.2 The Cubic Facet Model

In the cubic facet model [42, 43], it is assumed that in each neighborhood of an image, the signal can be described by a cubic polynomial

$$\begin{aligned} f(x,y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 \\ + k_7 x^3 + k_8 x^2 y + k_9 x y^2 + k_{10} y^3. \end{aligned} \tag{3.39}$$

The coefficients $\{k_i\}$ are determined by an unweighted least squares fit within a square window of some size. A typical application of the cubic facet model is to estimate the image derivatives from the polynomial model and to use these to get the curvature

$$\kappa = \frac{f_x^2 f_{yy} + f_y^2 f_{xx} - 2 f_x f_y f_{xy}}{(f_x^2 + f_y^2)^{3/2}} = \frac{2(k_2^2 k_6 + k_3 k_4 - k_2 k_3 k_5)}{(k_2^2 + k_3^2)^{3/2}}. \tag{3.40}$$

We see that the cubic facet model has much in common with normalized convolution, except that it lacks provision for certainty and applicability. Hence we can regard this model as a special case of normalized convolution, with third degree polynomials as basis functions, certainty identically one, and applicability identically one on a square window. We can also note that in the computation of the curvature by equation (3.40), some of the estimated coefficients are not used, which can be compared to the idea of normalized differential convolution, section 3.6.

Facet models in general[11] can of course also be described in terms of normalized convolution, by changing the set of basis functions accordingly. Applications for the facet model include gradient edge detection, zero-crossing edge detection, image segmentation, line detection, corner detection, three-dimensional shape estimation from shading, and determination of optic flow [43]. By extension, the same approaches can be taken with normalized convolution and probably with

[10] Notice that by equation (3.30), the equivalent correlation kernel when we have constant certainty is given by a multiple of the applicability, since we have only one basis function, which is a constant.

[11] With other basis functions than the cubic polynomials.

better results, since the availability of the applicability mechanism allows better control of the process. As discussed in the following section, an appropriate choice of applicability is especially important if we want to estimate orientation[12]. The facet model is further discussed in section 4.8.2.

3.10 Choosing the Applicability

The choice of applicability depends very much on the application. It is in fact all but impossible to give general guidelines. For most applications, however, it seems more or less unavoidable that we wish to give higher importance to points in the center of the neighborhood than to points farther away. Thus the applicability should be monotonically decreasing in all directions. For an example of an exception to this rule, see [56].

Another property to be aware of is isotropy. Unless a specific direction dependence is wanted, one probably had better taking care to get an isotropic applicability. This is, in fact, of utmost importance in the orientation estimation algorithm presented in chapter 5, see in particular section 5.7.1.

If we look at a specific application, the normalized averaging from section 3.9.1, we can see a trade-off between excessive blurring with a wide applicability function and noise caused by the varying certainty with a narrow applicability. The motivation for the very narrow applicability in figure 3.4(a) is that we want to interpolate from values as close as possible to the point of interest and more or less ignore information farther away. In other applications it is necessary to have a wider applicability, because we actually want to analyze a whole neighborhood, e.g. to estimate orientation. In these cases *the size of the applicability is related to the scale of the analyzed features*. Another reason for a wider applicability is to become less sensitive to signal noise.

3.11 Towards a Statistical Interpretation of Certainty

As defined in section 3.2, certainty is only characterized by how it affects the weighting in the least squares problems (3.2) or (3.3). It would be useful if we could relate certainty to the statistical variations of a measured signal. We do not have a solution to this problem, but there is an interesting structural similarity to the following classical result from statistics [80].

Assume $\mathbf{f} = \mathbf{B}\mathbf{r} + \mathbf{e}$, where $\mathbf{e}$ is a stochastic variable with zero mean and covariance matrix $\mathbf{V}$. We want to find the best linear unbiased estimate $\hat{\mathbf{r}}$ of $\mathbf{r}$, i.e. $\hat{\mathbf{r}} = \mathbf{L}\mathbf{f}$, $E[\mathbf{r} - \hat{\mathbf{r}}] = \mathbf{0}$, and $E[(\mathbf{r} - \hat{\mathbf{r}})(\mathbf{r} - \hat{\mathbf{r}})^T]$ is minimized. This is given by

$$\hat{\mathbf{r}} = (\mathbf{B}^T\mathbf{V}^{-1}\mathbf{B})^{-1}\mathbf{B}^T\mathbf{V}^{-1}\mathbf{f}. \tag{3.41}$$

[12]Notice in particular the dramatic difference between a square applicability, implicitly used by the facet model, and a Gaussian applicability in table 5.3.

Comparing (3.41) to (3.7) seems to indicate that we should choose the certainty so that

$$\mathbf{W}_a\mathbf{W}_c = \mathbf{V}^{-1} \tag{3.42}$$

holds in all neighborhoods. There are two difficulties, however. The first one is that $\mathbf{V}$ cannot be determined only from measurement errors in the signal $\mathbf{f}$. It is also necessary to include the errors implicit in the signal model $\mathbf{f} = \mathbf{B}\mathbf{r}$. The second one is that we have no solution to (3.42) if $\mathbf{V}$ is not diagonal.

3.12 Further Generalizations of Normalized Convolution

In the formulation of normalized convolution, it is traditionally assumed that the local signal model is spanned by a set of vectors constituting a subspace basis. As we have already discussed in section 3.2, this assumption is not without complications, since the vectors may effectively become linearly dependent in the seminorm given by $\mathbf{W}$. This typically happens in areas with large amounts of missing data. A first generalization is therefore to allow linearly dependent vectors in the signal model, i.e. exchanging the subspace basis for a subspace frame. Except that we lose the simplifications to the expressions (3.7) and (3.8), this case has already been covered by the presentation in section 3.2.

With a subspace frame instead of a subspace basis, another possible generalization is to use a weighted norm $\mathbf{L}$ in the coefficient space instead of the Euclidian norm, i.e. generalizing the seminorm weighted general linear least squares problem (3.2) somewhat to

$$\arg\min_{\mathbf{r}\in\mathcal{S}} \|\mathbf{r}\|_{\mathbf{L}}, \qquad \mathcal{S} = \{\mathbf{r} \in \mathcal{C}^m; \|\mathbf{B}\mathbf{r} - \mathbf{f}\|_{\mathbf{W}} \text{ is minimum}\}. \tag{3.43}$$

If we require $\mathbf{L}$ to be positive definite, the solution is now given by (2.61) as

$$\mathbf{r} = \mathbf{L}^{-1}(\mathbf{W}\mathbf{B}\mathbf{L}^{-1})^{\dagger}\mathbf{W}\mathbf{f} \tag{3.44}$$

or by (2.62) as

$$\mathbf{r} = \mathbf{L}^{-1}(\mathbf{L}^{-1}\mathbf{B}^{*}\mathbf{W}^2\mathbf{B}\mathbf{L}^{-1})^{\dagger}\mathbf{L}^{-1}\mathbf{B}^{*}\mathbf{W}^2\mathbf{f}. \tag{3.45}$$

If we allow $\mathbf{L}$ to be semidefinite we have to resort to the solution given by (2.60).

If $\mathbf{L}$ is diagonal, the elements can be interpreted as the relative cost of using each subspace frame vector. This case is not very interesting, however, since the same effect could have been achieved simply by an amplitude scaling of the frame vectors. A more general $\mathbf{L}$ allows varying the costs for specific linear combinations of the subspace frame vectors, leading to more interesting possibilities.

That it would be pointless to introduce $\mathbf{L}$ in the case of a subspace basis is clear from section 2.4.3, since it would not affect the solution at all, unless we have the case where the seminorm $\mathbf{W}$ turns the basis vectors effectively linearly dependent. Correspondingly, it does not make much sense to use a basis or a frame for the whole space as signal model, since in this case the weighting by $\mathbf{W}$ would be superfluous as the error to be minimized in (3.2) would be zero regardless of

norm. Hence neither the certainty nor the applicability would make a difference to the solution.

Another generalization that could be solved by the framework from chapter 2 is to have a non-diagonal weight matrix $\mathbf{W}$. It is not clear how to interpret this but it is possible that the certainty part $\mathbf{W}_c$ could naturally be non-diagonal if the primary measurements of the signal were collected, e.g., in the frequency domain or as line integrals. See also section 3.11.

A different generalization, that is not covered by the framework from the previous chapter, is to replace the general linear least squares problem (3.2) with the simultaneous minimization of signal error and coefficient norm,

$$\arg\min_{\mathbf{r}} \alpha\|\mathbf{Br}-\mathbf{f}\|_{\mathbf{W}} + \beta\|\mathbf{r}\|. \tag{3.46}$$

This approach could possibly be more robust when the basis functions are nearly linearly dependent, but we will not investigate it further here.

A generalization which allows arbitrary positioning of sample points, called continuous normalized convolution, has recently been presented by Andersson [3, 5].

Chapter 4

Polynomial Expansion

4.1 Introduction

Instead of working directly on the image intensity values, most image analysis methods start with some, often linear, transformation of the data. Examples of this include Fourier transforms, Wavelet transforms, and all kinds of filtering, e.g. a Gabor filter bank. This chapter introduces a transformation we call polynomial expansion, which is the basis for the orientation estimation algorithm in chapter 5 and the displacement estimation algorithm in chapter 7.

The basic idea of polynomial expansion is to approximate a neighborhood of each pixel with a polynomial. For this to be useful it is assumed that the polynomial coefficients capture sufficient information about the signal. Polynomials of any degree can be used, but in this thesis the primary interest is on quadratic polynomials. Then the local DC level is captured by the constant term, the odd part of the signal by the linear term, and the even part of the signal by the quadratic term.

Readers who are familiar with Haralick's facet model [41, 42, 43, 44] undoubtedly will notice that polynomial expansion looks very similar. They do have elements in common but for reasons explained in section 4.8.2 we consider this method sufficiently dissimilar to be called by a different name. In particular, being based on normalized convolution, this method allows uncertain signal values and *weighted* least squares fitting. The latter is a very important feature, as demonstrated in section 5.7.

Applications of polynomial expansion include orientation and displacement estimation, as described later in this thesis, and detection of rotational symmetries [54]. Additionally most of the applications of the cubic facet model should work at least as well with polynomial expansion, e.g. gradient edge detection, zero-crossing edge detection, image segmentation, line detection, corner detection, three-dimensional shape estimation from shading, and determination of optic flow [43].

4.2 Estimating the Coefficients of a Polynomial Model

To simplify the presentation we exclusively study expansion into quadratic polynomials and comment on the generalization to other degree polynomials in section 4.7. Thus we have the local signal model, expressed in a local coordinate system,

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c, \tag{4.1}$$

where $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ a vector and c a scalar. The coefficients of the model can be estimated in terms of normalized convolution with the basis functions

$$\{1, x, y, x^2, y^2, xy\} \tag{4.2}$$

for the 2D case and obvious generalizations to higher dimensionalities. The relation between the coefficients $\{r_i\}$ obtained from normalized convolution and the signal model (4.1) is straightforward. In 2D we have

$$c = r_1, \quad \mathbf{b} = \begin{pmatrix} r_2 \\ r_3 \end{pmatrix}, \text{ and } \mathbf{A} = \begin{pmatrix} r_4 & \frac{r_6}{2} \\ \frac{r_6}{2} & r_5 \end{pmatrix}, \tag{4.3}$$

so that

$$\begin{pmatrix} x & y \end{pmatrix} \mathbf{A} \begin{pmatrix} x \\ y \end{pmatrix} + \mathbf{b}^T \begin{pmatrix} x \\ y \end{pmatrix} + c = r_1 + r_2 x + r_3 y + r_4 x^2 + r_5 y^2 + r_6 xy. \tag{4.4}$$

The choice of applicability depends on the application. In the context of orientation estimation isotropy is an important property, as we will see in sections 5.5 and 5.7. More generally this holds for any application where we wish to avoid a directional bias. For certain other applications, on the other hand, anisotropy may be a desired feature. When it comes to performance it is very advantageous to have a Cartesian separable applicability, as is shown in section 4.3. The size of the applicability determines the scale of the structures captured by the polynomial expansion. This is further discussed in section 4.5.

We can notice that $\mathbf{A}$ captures information about the even part of the signal, excluding DC, that $\mathbf{b}$ captures information about the odd part of the signal, and that c varies with the local DC level. In the applications presented in the following chapters c does not give any useful information[1] but is still necessary to include in the signal model because otherwise the DC level would affect the computation of $\mathbf{A}$ and sometimes also $\mathbf{b}$.

The use of normalized convolution allows us to have signals with varying certainty, but we are not going to explore this case in full depth. In particular we will assume that the basis functions are always effectively linearly independent[2] so that we can use equation (3.8) to compute normalized convolution. As noted in

[1] It may do in other applications.

[2] Notice that even with constant certainty this requires the applicability not to be too small. E.g. in 2D an applicability with only five non-zero values can never be sufficient for the six basis functions.

section 3.2.4, that equation can also be used for continuous signals if we introduce a suitable inner product. Here we use the standard L^2 inner product

$$(f, g) = \int_{\mathcal{R}^N} f(\mathbf{x}) g(\mathbf{x})\, d\mathbf{x}. \tag{4.5}$$

Although we do not limit ourselves to L^2 functions, it is assumed that all integrals are convergent, typically by requiring that the applicability has finite support or decreases exponentially while the basis functions and the signals are bounded by some polynomial.

4.3 Fast Implementation

A drawback with normalized convolution is that it is computationally demanding. If we assume that the certainty is constant, however, it can be reduced to ordinary convolution or cross correlation[3], as shown in section 3.7. To further improve on the computational complexity, it turns out that the resulting correlation kernels are Cartesian separable for suitable choices of applicability.

4.3.1 Equivalent Correlation Kernels

To find the equivalent correlation kernels, assuming constant certainty, we need to compute the dual basis functions according to equation (3.30),

$$\begin{pmatrix} | & | & & | \\ \tilde{\mathbf{b}}_1 & \tilde{\mathbf{b}}_2 & \dots & \tilde{\mathbf{b}}_m \\ | & | & & | \end{pmatrix} = \begin{pmatrix} | & | & & | \\ \mathbf{a}\cdot\mathbf{b}_1 & \mathbf{a}\cdot\mathbf{b}_2 & \dots & \mathbf{a}\cdot\mathbf{b}_m \\ | & | & & | \end{pmatrix} \mathbf{G}^{-1}, \tag{4.6}$$

where

$$\mathbf{G} = \begin{pmatrix} (\mathbf{a}\cdot\mathbf{b}_1, \mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{b}_1, \mathbf{b}_m) \\ \vdots & \ddots & \vdots \\ (\mathbf{a}\cdot\mathbf{b}_m, \mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{b}_m, \mathbf{b}_m) \end{pmatrix}. \tag{4.7}$$

Now equation (3.29) tells us that we get the coefficients in the signal model (4.1) by the cross correlation

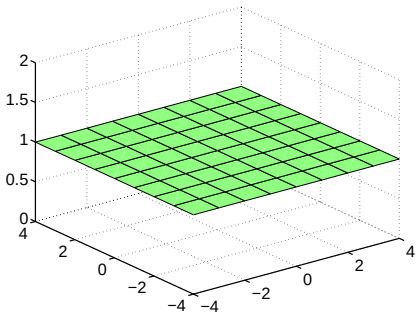
$$r_i(\mathbf{x}) = (\tilde{\mathbf{b}}_i \star f)(\mathbf{x}), \tag{4.8}$$

where we can skip $i = 1$ in applications which do not make use of the DC level.

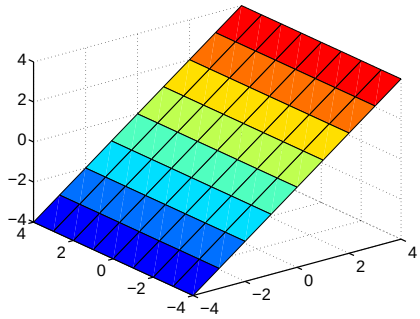
To illustrate these equivalent correlation kernels, we show the six basis functions for 2D in figure 4.1 and in figure 4.2 the dual basis functions for a Gaussian applicability, $a(\mathbf{x}) = e^{-0.5\mathbf{x}^T\mathbf{x}}$, on a 9×9 grid. In figure 4.3 finally we have the Fourier transforms of the equivalent correlation kernels[4].

[3] We prefer to use the latter operation, but remember that a real correlation kernel always can be transformed into a convolution kernel simply by reflecting it.

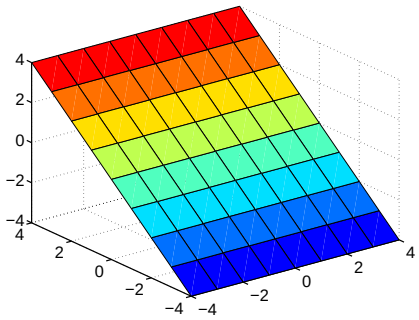
[4] See also section 4.8.1 for an interpretation of the dual basis functions when the applicability is Gaussian.



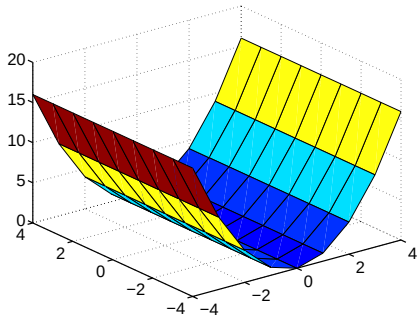
(a) 1



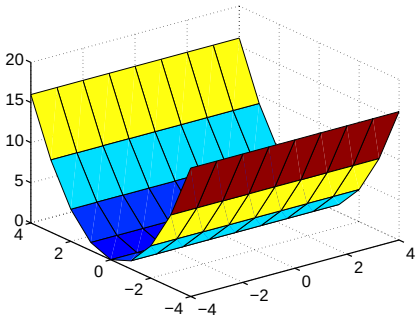
(b) x



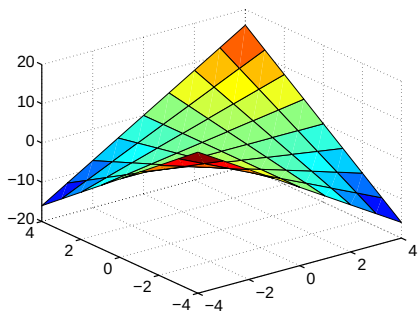
(c) y



(d) x^2

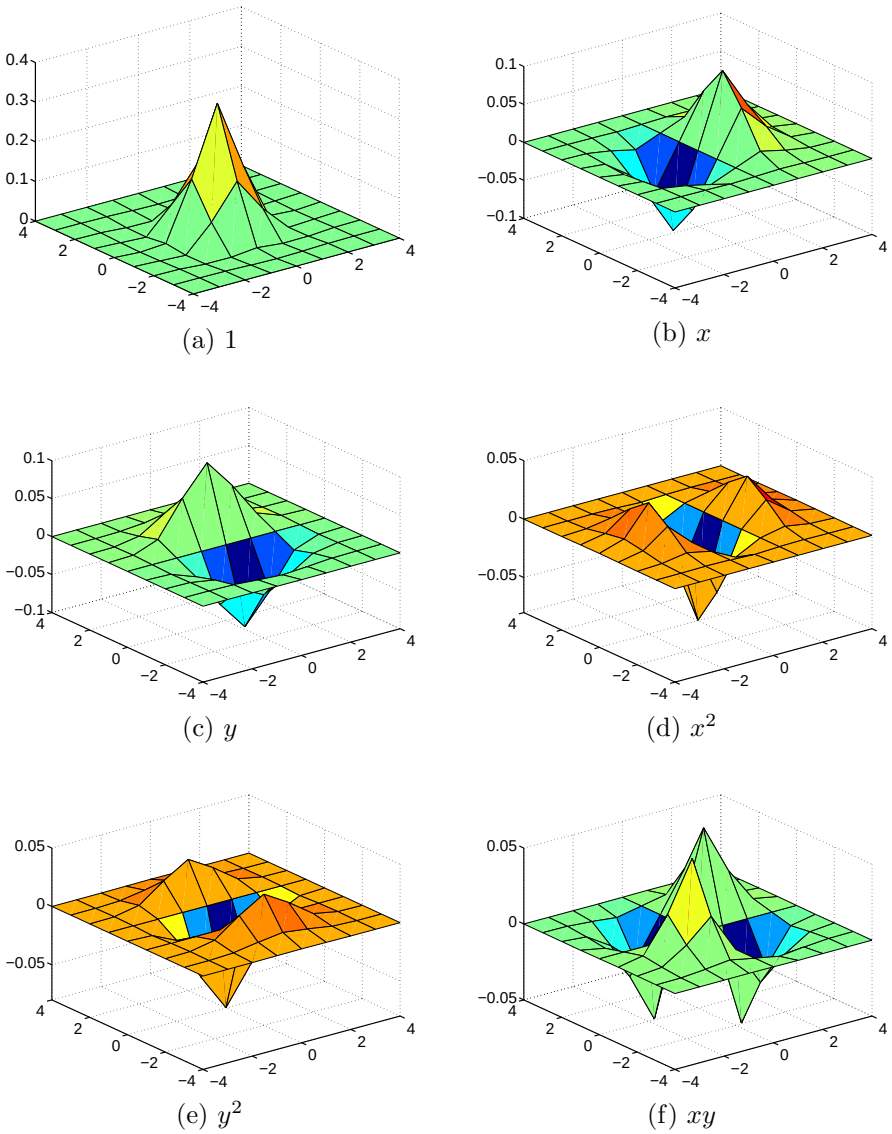


(e) y^2



(f) xy

Figure 4.1: Basis functions in 2D.

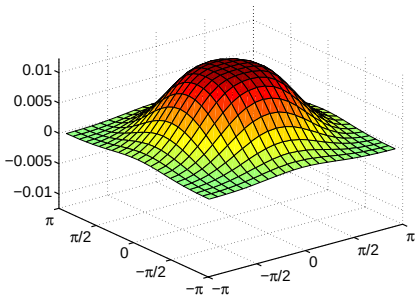


(a) 1 (b) x

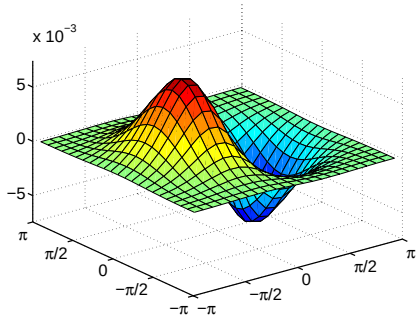
(c) y (d) x^2

(e) y^2 (f) xy

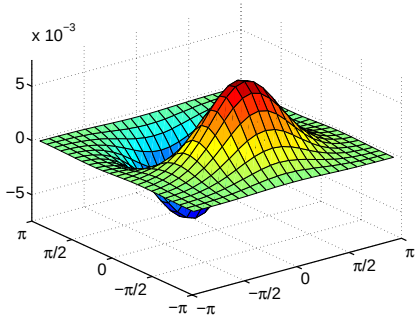
Figure 4.2: Dual basis functions in 2D.



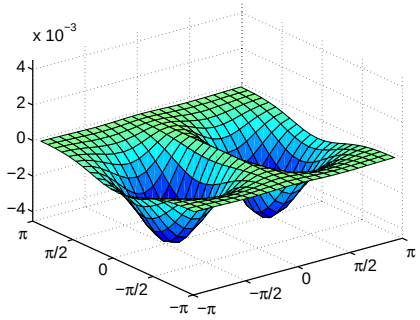
(a) 1 (real part)



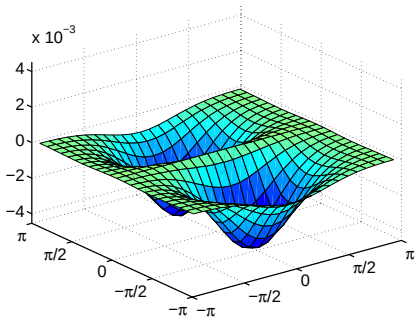
(b) x (imaginary part)



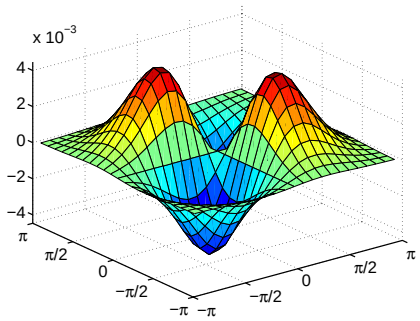
(c) y (imaginary part)



(d) x^2 (real part)



(e) y^2 (real part)



(f) xy (real part)

Figure 4.3: Fourier transforms of equivalent correlation kernels.

4.3.2 Cartesian Separability

It turns out that all the correlation kernels in figure 4.2, except the somewhat less interesting one corresponding to the constant basis function, have the property that they are Cartesian separable, i.e. that they can each be decomposed as the outer product of two 1D kernels, one horizontal and one vertical. This means that each cross correlation can be computed by means of two consecutive 1D cross correlations, which computationally is significantly more efficient than a full 2D cross correlation. This advantage is even more important for higher dimensionalities than two.

If we have a Cartesian separable applicability we can see that the products $\{\mathbf{a} \cdot \mathbf{b}_k\}$ in equation (4.6) also have that property, because the basis functions obviously are Cartesian separable. This means that the polynomial coefficients, for signals of any dimensionality, can be computed solely by 1D correlations, since (4.6) and (4.8) together give us

$$\mathbf{r}(\mathbf{x}) = \mathbf{G}^{-1} \begin{pmatrix} ((\mathbf{a} \cdot \mathbf{b}_1) \star \mathbf{f})(\mathbf{x}) \\ \vdots \\ ((\mathbf{a} \cdot \mathbf{b}_m) \star \mathbf{f})(\mathbf{x}) \end{pmatrix}. \tag{4.9}$$

The next step is to explore the structures of $\mathbf{G}$ and $\mathbf{G}^{-1}$. It turns out that they become extremely simple if we restrict ourselves to applicabilities which are even and identical along all axes, i.e. invariant under reflection and permutation of the axes. Then most of the inner products $\{(\mathbf{a} \cdot \mathbf{b}_k, \mathbf{b}_l)\}$ in equation (4.7) become zero since most of the products $\{\mathbf{a} \cdot \mathbf{b}_k \cdot \mathbf{b}_l\}$ are odd along at least one axis. In fact, the only non-zero inner products are $\{(\mathbf{a} \cdot \mathbf{b}_k, \mathbf{b}_k)\}$, $\{(\mathbf{a} \cdot \mathbf{b}_1, \mathbf{b}_{x_i^2})\}$, and $\{(\mathbf{a} \cdot \mathbf{b}_{x_i^2}, \mathbf{b}_{x_j^2})\}$. Thus we have the structure of $\mathbf{G}$, illustrated in the 3D case,

$$\mathbf{G} = \begin{pmatrix} a & & & & b & b & b & & & & 1 \\ & b & & & & & & & & & x \\ & & b & & & & & & & & y \\ & & & b & & & & & & & z \\ b & & & & c & d & d & & & & x^2 \\ b & & & & d & c & d & & & & y^2 \\ b & & & & d & d & c & & & & z^2 \\ & & & & & & & d & & & xy \\ & & & & & & & & d & & xz \\ & & & & & & & & & d & yz \\ 1 & x & y & z & x^2 & y^2 & z^2 & xy & xz & yz & \end{pmatrix}. \tag{4.10}$$

Surprisingly enough we get an even simpler structure for the inverse,

$$\mathbf{G}^{-1} = \begin{pmatrix} \mathfrak{a} & & & & \mathfrak{e} & \mathfrak{e} & \mathfrak{e} & & & \\ & \mathfrak{b} & & & & & & & & \\ & & \mathfrak{b} & & & & & & & \\ & & & \mathfrak{b} & & & & & & \\ \mathfrak{e} & & & & \mathfrak{c} & & & & & \\ \mathfrak{e} & & & & & \mathfrak{c} & & & & \\ \mathfrak{e} & & & & & & \mathfrak{c} & & & \\ & & & & & & & \mathfrak{d} & & \\ & & & & & & & & \mathfrak{d} & \\ & & & & & & & & & \mathfrak{d} \end{pmatrix} \begin{matrix} 1 \\ x \\ y \\ z \\ x^2 \\ y^2 \\ z^2 \\ xy \\ xz \\ yz \end{matrix} \quad . \tag{4.11}$$
$$\begin{matrix} 1 & x & y & z & x^2 & y^2 & z^2 & xy & xz & yz \end{matrix}$$

This result is proved in appendix A.

Now we can see that all the dual basis functions except the first one[5] are indeed separable. By (4.9) and (4.11) the duals can be written as

$$\begin{aligned} \tilde{\mathbf{b}}_1 &= \mathbf{a} \cdot (\mathfrak{a}\, \mathbf{b}_1 + \mathfrak{e} \sum \mathbf{b}_{x_i^2}), \\ \tilde{\mathbf{b}}_{x_i} &= \mathfrak{b}\, \mathbf{a} \cdot \mathbf{b}_{x_i}, \\ \tilde{\mathbf{b}}_{x_i^2} &= \mathbf{a} \cdot (\mathfrak{e}\, \mathbf{b}_1 + \mathfrak{c}\, \mathbf{b}_{x_i^2}) = \beta\, \mathbf{a} \cdot (\mathbf{b}_{x_i^2} - \alpha\, \mathbf{b}_1), \\ \tilde{\mathbf{b}}_{x_i x_j} &= \mathfrak{d}\, \mathbf{a} \cdot \mathbf{b}_{x_i x_j}, \quad i \neq j, \end{aligned} \tag{4.12}$$

where the constant α turns out to have precisely the value which makes the DC response of those dual basis functions zero. This should come as no surprise since we included the constant basis function precisely for this purpose.

The final step to get an efficient correlator structure is to notice that the decompositions into 1D correlation kernels have a lot of common factors. Figure 4.4 shows how the correlations for 3D can be structured hierarchically in three levels, where the first level contains correlations in the x direction, the second in the y direction, and the third in the z direction. The results are the correlations $\{((\mathbf{a} \cdot \mathbf{b}_k) \star \mathbf{f})(\mathbf{x})\}$ and the desired polynomial coefficients are then computed from equation (4.9) using the precomputed $\mathbf{G}^{-1}$ according to equation (4.11). It should be noted that only three different correlation kernels are in use, $\mathbf{a}_x$, $\mathbf{a}_x \cdot x$, and $\mathbf{a}_x \cdot x^2$ in the x direction and identical kernels in the other directions. These kernels are illustrated in figure 4.5. It should also be clear that this correlator structure straightforwardly can be extended to any dimensionality.

This scheme can be improved somewhat by replacing x^2 with $x^2 - \alpha$, doing the same for the other squares, and then, if the DC level is not needed, removing the leftmost box of the bottom level. Additionally the remaining coefficients in $\mathbf{G}^{-1}$ could be multiplied into the bottom level kernels to save another few multiplications. We do not consider these improvements in the complexity analysis in the following section.

[5] Notice that we always have the constant function as $\mathbf{b}_1$. Thus there is no ambiguity whether the subscript refers to the position number or to the zeroth degree monomial.

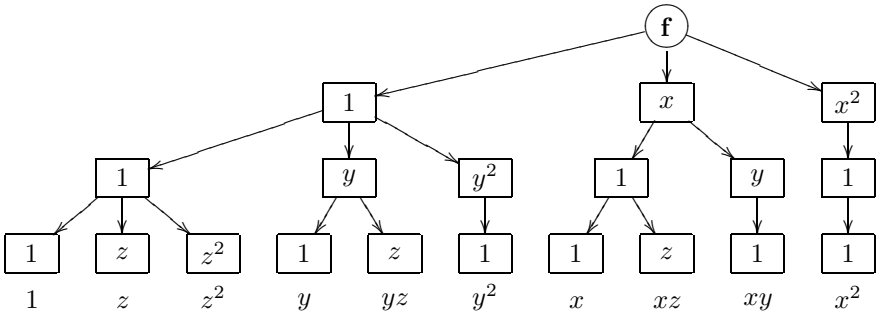


Figure 4.4: Correlator structure. There is understood to be an applicability factor in each box as well.

Finally one may ask oneself whether there exist applicabilities which are simultaneously Cartesian separable and isotropic. Obviously the Gaussians, $e^{-\rho \mathbf{x}^T \mathbf{x}}$, satisfy these requirements. This is the only solution, which is proved in appendix B.

4.4 Computational Complexity

The computational complexity of polynomial expansion depends on a number of factors, such as

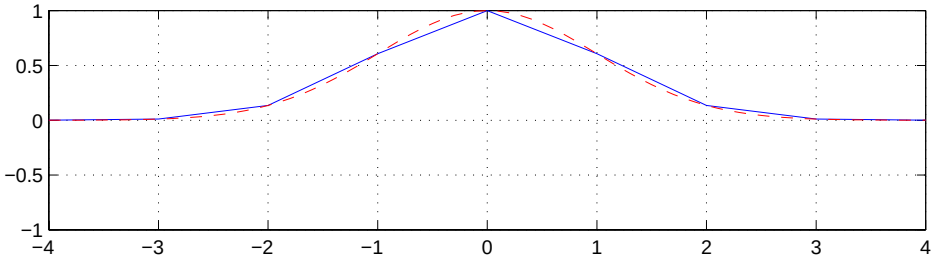
- the dimensionality d of the signal space,
- the size of the applicability per dimension, n,
- whether the certainty is assumed to be constant, and
- whether the applicability is separable and sufficiently symmetric.

Obviously it also depends on the degree of the polynomial signal model, but here we continue to restrict ourselves to quadratic polynomials.

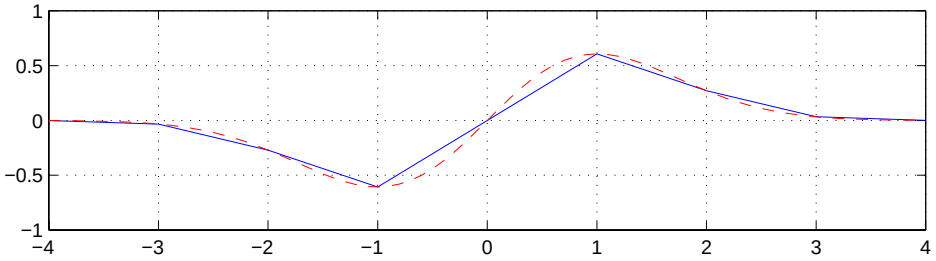
We consider the following four variations of the method:

Normalized Convolution (NC) The certainty is allowed to vary and the applicability is arbitrary. The polynomial coefficients are computed by equation (3.9), using the point by point strategy. It should be noticed that there are a number of duplicates among the quantities $\{\mathbf{a} \cdot \mathbf{b}_k \cdot \mathbf{b}_l\}$, reducing the number of inner products that must be computed.

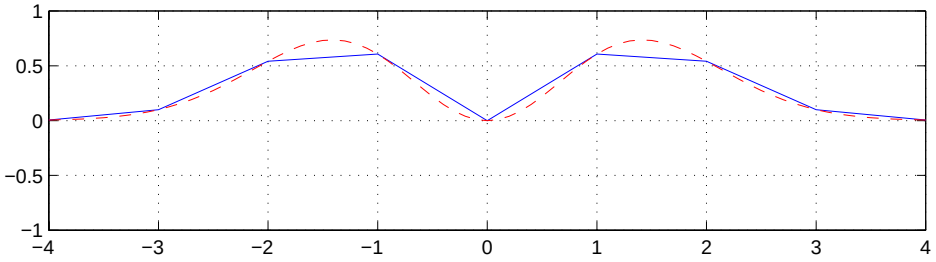
Correlation (C) The certainty is assumed to be constant (ignoring border effects) while the applicability is arbitrary. The polynomial coefficients are computed by equation (4.8).



(a) $\mathbf{a}_x \cdot 1$



(b) $\mathbf{a}_x \cdot x$



(c) $\mathbf{a}_x \cdot x^2$

Figure 4.5: One dimensional correlation kernels.

Method	Time complexity	Memory overhead
NC	$\frac{d^4}{24}n^d$	0
C	$\frac{d^2}{2}n^d$	0
SC	$\frac{d^3}{6}n$	1
SNC	$\frac{d^5}{120}n + \frac{d^6}{48}$	$\frac{d^4}{24}$

Table 4.1: Asymptotic complexities, d and n large, leading terms.

Separable Correlation (SC) The certainty is assumed to be constant and the applicability to be Cartesian separable and sufficiently symmetric. The polynomial coefficients are computed by the correlator structure in figure 4.4 followed by multiplication with $\mathbf{G}^{-1}$, having the structure (4.11).

Separable Normalized Convolution (SNC) With varying certainty but Cartesian separable applicability, all inner products in equation (3.9) can be computed as separable correlations. They can in fact even be computed hierarchically with a more complex variation of the structure in figure 4.4. The 2D case is illustrated in appendix C.

Independent of method we always have $m = \frac{(d+1)(d+2)}{2}$ basis functions. We count the complexity per computed set of expansion coefficients and only the number of multiplications involved; usually there is a slightly lower number of additions as well. This is consistent with the traditional count of coefficients for convolution kernels. Without going into details we present the asymptotic complexities, for both d and n large, in table 4.1. Memory overhead should be multiplied by the number of neighborhoods for which expansion coefficients are computed, to get the necessary size of temporary storage, measured in floating point values of the desired precision.

Usually, however, we are more interested in small values of d rather than in large values. A more detailed estimation of the complexity for 2D, 3D, and 4D can be found in table 4.2. The values relate to reasonably straightforward implementations of the methods and can likely be improved somewhat. The first term of the time complexities is the total number of coefficients involved in the correlation kernels, while the second term is the count for the transformation from correlation results to expansion coefficients. Included in the latter part for NC and SNC is the solution of an $m \times m$ symmetric positive definite equation system, estimated at $\frac{m^3}{6} + \frac{3m^2}{2} + \frac{m}{3}$ operations [77].

To sum this analysis up, it is clear that separable correlation is by far the most efficient method. The restriction set on the applicability is no severe limitation because those properties are usually desired anyway. The requirement of constant certainty is a problem, however, since such an assumption surely fails at least in a neighborhood of the borders and the method is more than likely to yield significantly biased results there. Proper attention to the vanishing certainty outside the border is paid by the NC and SNC methods, which on the other hand have a

	Time complexity			Memory overhead		
Method	$d = 2$	$d = 3$	$d = 4$	$d = 2$	$d = 3$	$d = 4$
NC	$21n^2 + 92$	$45n^3 + 320$	$85n^4 + 905$	0	0	0
C	$6n^2$	$10n^3$	$15n^4$	0	0	0
SC	$9n + 10$	$19n + 16$	$34n + 23$	1	1	1
SNC	$29n + 92$	$74n + 320$	$159n + 905$	16	36	71

Table 4.2: Time complexity and memory overhead for 2D, 3D, and 4D.

high time complexity and a large memory overhead, respectively. A good solution for signals with constant certainty would be to use separable correlation for the central part of the signal and normalized convolution or separable normalized convolution for the border parts. It must be stressed, however, that while normalized convolution will reduce the negative impact of missing information outside the borders, it will certainly not remove it completely. The best solution is, as always, to keep away from those parts as much as possible.

4.5 Polynomial Expansion in Multiple Scales

As was noted in section 3.10, the size of the applicability is related to the scale of the analyzed features. Thus it may be useful to do polynomial expansion in multiple scales. If speed issues are disregarded, this is trivially implemented by doing multiple polynomial expansions with a sequence of scaled applicabilities, e.g. a sequence of Gaussians with varying standard deviations.

In most cases, however, we cannot disregard speed. At coarse scales the applicabilities tend to become very large, and even if we are using one of the separable methods, the computations may become too slow. A computationally less expensive solution is to first compute a lowpass pyramid from the signal and apply polynomial expansion to this. This is particularly efficient computationally if the lowpass pyramid includes subsampling.

The question is how these approaches relate to each other. We can analyze this exactly in a special case. Assume that we apply quadratic polynomial expansion with Gaussian applicability to a signal which has passed a Gaussian lowpass filter. We do this continuously in 1D for the case of constant certainty. Let

$$g_1(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{x^2}{2\sigma_1^2}}, \tag{4.13}$$

$$g_2(x) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{x^2}{2\sigma_2^2}}, \tag{4.14}$$

$$g_3(x) = \frac{1}{\sqrt{2\pi}\sigma_3} e^{-\frac{x^2}{2\sigma_3^2}}, \tag{4.15}$$

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2. \tag{4.16}$$

The original signal f is lowpass filtered by correlation with g_1, giving $g_1 \star f$.

Polynomial expansion of this signal with applicability g_2 gives by (4.7), (4.9), and results from appendix D,

$$\begin{aligned}
\mathbf{r}'(x) &= \begin{pmatrix} (g_2,1) & (g_2,x) & (g_2,x^2) \\ (g_2,x) & (g_2,x^2) & (g_2,x^3) \\ (g_2,x^2) & (g_2,x^3) & (g_2,x^4) \end{pmatrix}^{-1} \begin{pmatrix} (g_2 \star (g_1 \star f))(x) \\ (xg_2 \star (g_1 \star f))(x) \\ (x^2 g_2 \star (g_1 \star f))(x) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & \sigma_2^2 \\ 0 & \sigma_2^2 & 0 \\ \sigma_2^2 & 0 & 3\sigma_2^4 \end{pmatrix}^{-1} \begin{pmatrix} (g_3 \star f)(x) \\ \frac{\sigma_2^2}{\sigma_3^2}(xg_3 \star f)(x) \\ \frac{\sigma_2^4}{\sigma_3^4}(x^2 g_3 \star f)(x) + \frac{\sigma_1^2\sigma_2^2}{\sigma_3^2}(g_3 \star f)(x) \end{pmatrix} \\
&= \begin{pmatrix} \frac{3}{2} & 0 & -\frac{1}{2\sigma_2^2} \\ 0 & \frac{1}{\sigma_2^2} & 0 \\ -\frac{1}{2\sigma_2^2} & 0 & \frac{1}{2\sigma_2^4} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{\sigma_2^2}{\sigma_3^2} & 0 \\ \frac{\sigma_1^2\sigma_2^2}{\sigma_3^2} & 0 & \frac{\sigma_2^4}{\sigma_3^4} \end{pmatrix} \begin{pmatrix} (g_3 \star f)(x) \\ (xg_3 \star f)(x) \\ (x^2 g_3 \star f)(x) \end{pmatrix}.
\end{aligned} \tag{4.17}$$

If we instead apply polynomial expansion directly to f with g_3 as applicability, we obtain, still by (4.7), (4.9), and (D.22)–(D.26),

$$\begin{aligned}
\mathbf{r}(x) &= \begin{pmatrix} (g_3,1) & (g_3,x) & (g_3,x^2) \\ (g_3,x) & (g_3,x^2) & (g_3,x^3) \\ (g_3,x^2) & (g_3,x^3) & (g_3,x^4) \end{pmatrix}^{-1} \begin{pmatrix} (g_3 \star f)(x) \\ (xg_3 \star f)(x) \\ (x^2 g_3 \star f)(x) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & \sigma_3^2 \\ 0 & \sigma_3^2 & 0 \\ \sigma_3^2 & 0 & 3\sigma_3^4 \end{pmatrix}^{-1} \begin{pmatrix} (g_3 \star f)(x) \\ (xg_3 \star f)(x) \\ (x^2 g_3 \star f)(x) \end{pmatrix}.
\end{aligned} \tag{4.18}$$

Combining (4.17) and (4.18) we can express $\mathbf{r}'$ in terms of $\mathbf{r}$ as

$$\begin{aligned}
\mathbf{r}'(x) &= \begin{pmatrix} \frac{3}{2} & 0 & -\frac{1}{2\sigma_2^2} \\ 0 & \frac{1}{\sigma_2^2} & 0 \\ -\frac{1}{2\sigma_2^2} & 0 & \frac{1}{2\sigma_2^4} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{\sigma_2^2}{\sigma_3^2} & 0 \\ \frac{\sigma_1^2\sigma_2^2}{\sigma_3^2} & 0 & \frac{\sigma_2^4}{\sigma_3^4} \end{pmatrix} \begin{pmatrix} 1 & 0 & \sigma_3^2 \\ 0 & \sigma_3^2 & 0 \\ \sigma_3^2 & 0 & 3\sigma_3^4 \end{pmatrix} \mathbf{r}(x) \\
&= \begin{pmatrix} 1 & 0 & \sigma_1^2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{r}(x).
\end{aligned} \tag{4.19}$$

We see that the expansion coefficients are the same, with the exception of the DC coefficient, which gets an extra contribution from the quadratic coefficient. This result generalizes to higher dimensionalities. In summary polynomial expansion of a lowpass filtered signal is equivalent to a polynomial expansion, with a different applicability, of the original signal, with exception for the DC coefficient and under a number of fairly limiting assumptions.

Unfortunately we cannot lift any more assumptions. In the discrete case the approximation is good if σ_1 is small but not if we instead try to make σ_2 small. An evaluation of this in the context of orientation estimation can be found in section 5.7.2. For higher degree polynomials a similar relation can be derived but it becomes more complex. For other applicabilities and lowpass filters than

Gaussians the derivation is obviously not valid and it seems difficult to find similar relations.

However, applying polynomial expansion to a lowpass filtered signal is a perfectly reasonable operation, regardless whether it coincides with a polynomial expansion of the original signal for some applicability. What matters is whether it works well enough for the intended application.

Multi-scale computations also work well with the approximative methods in the following section. Those can naturally be extended to the case of non-constant certainty, using separate lowpass pyramids for c and cf.

4.6 Approximative Methods

The computational complexity of polynomial expansion can be reduced somewhat by using approximative methods, naturally at the cost of accuracy. In this section two such methods are presented, both limited to the important special case of Gaussian applicability.

4.6.1 Derivative Filters

Johansson [54] derives an approximative method from the observation that derivatives of Gaussians become polynomials times Gaussians, i.e.,

$$g(x) = e^{-\frac{x^2}{2\sigma^2}}, \tag{4.20}$$

$$g'(x) = -\frac{x}{\sigma^2} g(x), \tag{4.21}$$

$$g''(x) = \frac{x^2 - \sigma^2}{\sigma^4} g(x). \tag{4.22}$$

This makes it possible to replace the correlator structure 4.4 with a structure consisting of correlation with Gaussians followed by a sequence of partial differentiations. In the 2D case this looks like figure 4.6. The mapping from correlation outputs to expansion coefficients obviously becomes different from $\mathbf{G}^{-1}$ used in section 4.3 but is still linear.

The reason why this method is approximative is that the derivative relations (4.20)–(4.22) hold for the continuous case while we in reality have discretized Gaussians and discrete approximations of differentiation. Furthermore we want to keep the differentiation kernels small in order to improve speed but that is a trade-off against the quality of the result.

The reduction of the computational complexity as reported by Johansson for three cases is listed in table 4.3. Evaluation results for this method are included in section 5.7.4. For further information about the method, the reader is referred to [54].

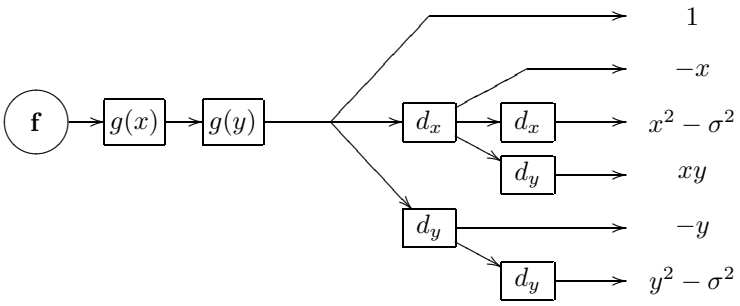


Figure 4.6: Correlator structure for approximative polynomial expansion in 2D with full certainty. From [54].

Method	Standard	Approximative
SC, 2D	$9n + 10$	$2n + 5m + 12$
SNC, 2D	$29n + 92$	$4n + 17m + 92$
SC, 3D	$19n + 16$	$3n + 9m + 22$

Table 4.3: Computational complexity for standard polynomial expansion and the approximative method using derivative filters. n is the size of the Gaussian kernel and m of the differentiation kernels.

4.6.2 Binomial Filters

Binomial filters are popular as approximations of Gaussians because they can be computed by cascading [1 1] filters. This allows an implementation with only additions, which on certain platforms can be very beneficial.

The way to make use of this for polynomial expansion is to let the applicability be binomial. This does not make it an approximative method per se, since the binomial is a perfectly valid applicability. It only becomes approximative when the binomial is considered as a replacement for a Gaussian applicability.

We construct a binomial applicability in 1D with $2n + 1$ binomial coefficients as

$$b_n(k) = \begin{cases} \binom{2n}{n+k}, & -n \leq k \leq n, \\ 0, & \text{otherwise}, \end{cases} \tag{4.23}$$

and extend to higher dimensionalities through Cartesian products. To obtain the highest possible speed we want to use the hierarchical correlator structure in figure 4.4. This involves the three correlation kernels $b_n(k)$, $kb_n(k)$, and $k^2b_n(k)$. Interestingly enough we have the identities, for $n > 0$,

$$b_n(k) = b_{n-1}(k-1) + 2b_{n-1}(k) + b_{n-1}(k+1), \tag{4.24}$$

$$kb_n(k) = nb_{n-1}(k-1) - nb_{n-1}(k+1), \tag{4.25}$$

$$k^2b_n(k) = n^2b_{n-1}(k-1) - 2n(n-1)b_{n-1}(k) + n^2b_{n-1}(k+1). \tag{4.26}$$

What this means is that the three correlation kernels can be factorized as the smaller binomial kernel b_{n-1} convolved with the kernels $[1\ 2\ 1]$, $[-n\ 0\ n]$, and $[n^2\ -2n(n-1)\ n^2]$ respectively. The identities (4.24)–(4.26) are proved in appendix E.

This property is obviously very useful for multi-scale polynomial expansion since the results of correlation with the b_n kernel can be reused in the following scale. Identities similar to (4.24)–(4.26) can be derived for higher exponents k^p too, allowing this approach to be used for the separable normalized convolution method as well, with the same multi-scale advantages.

The binomials have (at least) two disadvantages compared to Gaussians, however. The first one is that the scale of Gaussians can be varied continuously while binomials come at discrete sizes. Whether this is significant obviously depends on the application. The second disadvantage is that binomials, at least of moderate size, are not quite as isotropic as Gaussians. This can be seen in the comparison of different applicabilities in section 5.7.1.

4.7 Higher Degree Polynomial Expansion

In this thesis we primarily take an interest in quadratic polynomial expansion. There are no difficulties extending this to higher degree polynomials, however. The definition of polynomial expansion in terms of normalized convolution clearly applies and so does the computation of equivalent correlation kernels in the case of constant certainty. In the separable correlation method the products $\{\mathbf{a} \cdot \mathbf{b}_k\}$ remain separable. The $\mathbf{G}^{-1}$ matrix obtains a different structure, but remains fairly sparse. The hierarchical correlator structure in figure 4.4 generalizes straightforwardly and so does the corresponding structure in C.1 for separable normalized convolution.

If the dimensionality is d and the degree is n, the number of basis functions is $\binom{n+d}{d}$. This number increases rapidly with the degree, especially if the dimensionality is high, with consequences for the computational complexity. It may also be necessary to adapt the size of the applicability so that the basis functions do not become linearly dependent simply by being too many.

Of course polynomials of lower degree than two can be used as well, although degree zero reduces to normalized averaging, see section 3.9.1. It is also conceivable to use a basis consisting of monomials which do not span all polynomials of some degree, e.g. the bilinear basis $\{1, x, y, xy\}$ or the even degree basis $\{1, x^2, y^2, xy\}$.

4.8 Relation to Other Approaches

4.8.1 Relation to First and Second Derivatives

By the Maclaurin expansion, a sufficiently differentiable signal can in a neighborhood of the origin be expanded as

$$f(\mathbf{x}) = f(\mathbf{0}) + (\nabla f)^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathbf{H} \mathbf{x} + O(\|\mathbf{x}\|^3), \tag{4.27}$$

where the gradient ∇f contains the first derivatives of f at the origin and the Hessian $\mathbf{H}$ contains the second derivatives,

$$\nabla f = \begin{pmatrix} f_{x_1}(\mathbf{0}) \\ \vdots \\ f_{x_n}(\mathbf{0}) \end{pmatrix}, \qquad \mathbf{H} = \begin{pmatrix} f_{x_1x_1}(\mathbf{0}) & \dots & f_{x_1x_n}(\mathbf{0}) \\ \vdots & \ddots & \vdots \\ f_{x_nx_1}(\mathbf{0}) & \dots & f_{x_nx_n}(\mathbf{0}) \end{pmatrix}. \tag{4.28}$$

Clearly this expansion looks identical to the signal model (4.1) with $\mathbf{A} = \frac{1}{2}\mathbf{H}$, $\mathbf{b} = \nabla f$, and $c = f(0)$. There is, however, an important conceptual difference. The Maclaurin expansion is intended to be correct for an infinitesimal neighborhood, while our signal model is intended to approximate the signal over a larger neighborhood, specified by the applicability.

The Maclaurin expansion also has the principal problem that the mathematical definition of derivatives requires signal values arbitrarily close to the origin, which are not available for discretized signals. Another complication is that perfect differentiation would be extremely sensitive to noise. One way to get around this, which also allows computing derivatives at different scales, is to first convolve the signal with a Gaussian,

$$h = f * g, \quad g(\mathbf{x}) = e^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}} \tag{4.29}$$

and then differentiate the filtered signal h. By the laws of convolution, the partial derivatives of h can be computed as f convolved with the partial derivatives of g, which are known explicitly. In e.g. 2D we have

$$\begin{aligned} h_x &= f * g_x, & g_x &= -\frac{x}{\sigma^2}g, \\ h_y &= f * g_y, & g_y &= -\frac{y}{\sigma^2}g, \\ h_{xx} &= f * g_{xx}, & g_{xx} &= \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right)g, \\ h_{xy} &= f * g_{xy}, & g_{xy} &= \frac{xy}{\sigma^4}g, \\ h_{yy} &= f * g_{yy}, & g_{yy} &= \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right)g, \end{aligned} \tag{4.30}$$

and we can see that the structure of the partial derivatives of g agrees with the dual basis functions in (4.12) for a Gaussian applicability. These functions are also illustrated in figure 4.2.

We would like to stress, however, that this fact is purely coincidental and an effect of the special properties of the Gaussians. For other applicabilities we do not have this relation. Likewise we cannot start with an arbitrary filter set that implements some approximation of first and second derivatives and expect it to be equivalent to a polynomial expansion for some applicability. Still it may help the intuition to think of $\mathbf{A}$ and $\mathbf{b}$ in terms of image derivatives.

For higher degree polynomials the relation partly breaks down even for Gaussian applicability, in that the expansion coefficients become linear combinations of multiple partial derivatives. Already in the quadratic case this happens for the coefficient c, which becomes a linear combination of h, h_{xx}, and h_{yy}.

4.8.2 Comparison with Polynomial Facet Models

The facet model principle is that the observed signal can be considered as a noisy, discretized sampling of a distorted version of an underlying continuous or piecewise continuous function. In order to process the signal, models are introduced for both the underlying function and the noise and distortion, and parameters for the models are estimated [43].

This principle is very general and covers a wide area. When considering the facet model as a signal processing method we are of the opinion that one should also take into account the proposed computational techniques.

The most common facet models involve low degree polynomials, e.g. the cubic facet model,

$$\begin{aligned} f(x,y) = k_1 + k_2 x + k_3 y + k_4 x^2 + k_5 xy + k_6 y^2 \\ + k_7 x^3 + k_8 x^2 y + k_9 xy^2 + k_{10} y^3, \end{aligned} \tag{4.31}$$

which was also discussed in section 3.9.2. Following [43], the parameters are determined by solving an unweighted least squares problem over a rectangular neighborhood. To find the solution efficiently, the polynomial basis is transformed into an orthogonal one by the construction of discrete Chebyshev polynomials in 1D and a tensor product extension to higher dimensions. It is noticed that the coefficients in the orthogonal basis can be computed as linear combinations of the neighborhood values and that this property carries over to the coefficients in (4.31). The coefficients of the corresponding correlation kernels for the 5×5 neighborhood are listed but no explicit formulas are given. There is no comment about the computational advantages available from the separable construction of the orthogonal polynomials, but this is emphasized in a more recent publication [101].

If we compare this to cubic polynomial expansion, we see that the signal model is the same and that both methods use a least squares approach to determine the parameters. The computations can in both methods be done through correlation kernels or by a separable correlation scheme. The difference, which is very significant, is that the facet model method does not include any weighting of the least squares problem, neither through certainty nor through applicability. Although there is a comment in [43] that the mathematics for weighted least squares is similar to the mathematics for unweighted least squares, the presented theory does not suffice to make inclusion of weighting practically feasible, in particular not a weighting which varies over the signal like the certainty does.

The conclusion is that while polynomial facet models and polynomial expansion are similar at the outset, the inclusion of applicability and certainty, plus the explicit formulas for the correlation kernels, makes polynomial expansion into a substantially different method.

4.8.3 Relation to Polynomial Fit Filters

Burt [16] describes a surface interpolation method using moment images and polynomial fit filters. Moment images are defined in 1D as

$$I_p(i) = \sum_{m=-\infty}^{\infty} W(m) I(i+m) m^p, \tag{4.32}$$

where $I(i)$ are intensity values and $W(m)$ is a window function centered at $m = 0$. The polynomial fit filter is derived for the bilinear polynomial

$$P(m,n) = \alpha + \beta m + \gamma n + \delta mn \tag{4.33}$$

by minimizing

$$\mathrm{Err}(i,j) = \sum_{m,n=\frac{-(K-1)}{2}}^{\frac{K-1}{2}} W(m,n) S(i+m, j+n) \left(I(i+m, j+n) - P(m,n)\right)^2, \tag{4.34}$$

where $S(i,j)$ is a support image, not necessarily binary, and the filter width K is odd. The polynomial coefficients can be computed by setting the partial derivatives of (4.34) to 0 and solving the resulting equation system. Only the first coefficient is needed since the output of the filter is $G(i,j) = P(0,0) = \alpha$. The solution is given in terms of moment images as

$$G = \frac{I_{00}D - I_{10}E - I_{01}F}{S_{00}D - S_{10}E - S_{01}F}, \tag{4.35}$$

where

$$Ips(i,j) = \sum_{m,n} W(m,n) S(i+m, j+n) I(i+m, j+n) m^p n^s, \tag{4.36}$$

$$Sps(i,j) = \sum_{m,n} W(m,n) S(i+m, j+n) m^p n^s, \tag{4.37}$$

$$D = S_{20}S_{02} - S_{11}S_{11}, \tag{4.38}$$

$$E = S_{10}S_{02} - S_{01}S_{11}, \tag{4.39}$$

$$F = S_{01}S_{20} - S_{10}S_{11}. \tag{4.40}$$

It is also shown how these computations can efficiently be done in a scale pyramid for a certain class of window functions, and how the results apply to surface interpolation.

If we compare this approach to normalized convolution we can see similarities in that we have a signal I, certainties S, an applicability W, and that (4.34) is equivalent to equation (3.3). The main differences is that only polynomial basis functions are considered, only the coefficient corresponding to the constant basis function is computed, and that no general theory is developed.

The comparison to polynomial expansion is mostly the same. The idea of local weighted least squares fitting of a neighborhood to a polynomial is common but

only one coefficient is computed and only a specific polynomial model is considered. It is also interesting to analyze the solution (4.35). We recognize (4.36) as elements of the vector $\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{f}$ in equation (3.7) and (4.37) as elements of the matrix $\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B}$. Then (4.35) together with (4.38)–(4.40) give the Cramer's rule solution of the equation system for α. However, (4.35) does not give the correct solution for the stated signal model (4.33) but for $P(m,n) = \alpha + \beta m + \gamma n$.

Chapter 5

Orientation Estimation

5.1 Introduction

Orientation is a feature that fundamentally distinguishes multidimensional signals from one-dimensional signals, since the concept lacks meaning in the latter case. It is also a feature that is far from trivial both to represent and to estimate, as well as to define strictly for general signals.

The one case where there is no question how the orientation should be defined is for non-constant *simple* signals, i.e. signals that can be written as

$$f(\mathbf{x}) = h(\mathbf{x}^T\mathbf{n}) \tag{5.1}$$

for some non-constant function h of one variable and for some vector $\mathbf{n}$. This means that the function is constant on all hyper-planes perpendicular to $\mathbf{n}$ and we say that the signal is oriented in the direction of $\mathbf{n}$. Notice however that $\mathbf{n}$ is not unique in (5.1) since we could replace $\mathbf{n}$ by any multiple. Even if we normalize $\mathbf{n}$ to get the unit directional vector $\hat{\mathbf{n}}$, there is still an ambiguity between $\hat{\mathbf{n}}$ and $-\hat{\mathbf{n}}$. This ambiguity must be addressed by the representation of orientation.

Of course the class of globally simple signals is too restricted to be of much use, so we need to generalize the definition of orientation to more general signals. To begin with we notice that we usually are not interested in a global orientation. In fact it is understood throughout the rest of this thesis that by "orientation" we mean "local orientation", i.e. we only look at the signal behavior in some neighborhood of a point of interest. We can, however, still not rely on always having locally simple neighborhoods.

Unfortunately there is no obvious way in which to generalize the definition of orientation to non-simple signals. Assume for example that we have a signal composed as the sum of two simple signals with different orientations. Should the orientation now be some mean value of the two orientations, both orientations, or something else? To illustrate what kind of problems we have here we take a closer look at two examples. They are both two-dimensional and we are interested in the orientation at a neighborhood of the origin.

1. Let $f_1(x, y) = x$ and $f_2(x, y) = y$. Now the sum $f(x, y) = f_1(x, y) +$

$f_2(x, y) = x + y$ is simple as well, oriented in the $(1\ 1)^T$ direction.

2. Let $f_1(x, y) = x^2$ and $f_2(x, y) = y^2$. This time the sum $f(x, y) = f_1(x, y) + f_2(x, y) = x^2 + y^2$ is entirely isotropic and we cannot possibly prefer one direction over another.

In practice, exactly how we define the orientation of non-simple signals tends to be a consequence of how we choose to represent the orientation and what procedure we use to estimate it. In this presentation we represent orientation by tensors, essentially along the lines of Knutsson [40, 62], although with a somewhat different interpretation.

Tensors can be computed in multiple ways and still obtain about the same qualitative properties. The first method, independently proposed by Bigün and Granlund [9, 10] and Förstner and Gülch [33], computes the tensor as local averages of the outer product of gradients. The method by Knutsson computes them from quadrature filter responses, as briefly described in section 5.2. In this chapter we introduce a third method based on polynomial expansion of the signal.

These tensors are known in the literature both as *orientation* tensors [40] and *structure* tensors [53]. The reason for the second name is that they contain more information about the local strucure than only the dominant orientation. In 3D they can, e.g., distinguish between line structures and plane structures. Here we choose to use the name *orientation* tensors to signify that they really only contain information related to orientation, as opposed to, e.g., phase or frequency.

5.2 The Orientation Tensor

In this section we give an overview of Knutsson's orientation tensor representation and estimation by means of quadrature filter responses [40, 62]. It should be noted that this estimation method is included only for reference and comparison. The estimation method used in this thesis is described in section 5.4.

5.2.1 Representation of Orientation for Simple Signals

The orientation tensor is a representation of orientation that for N-dimensional signals takes the form of an $N \times N$ real symmetric matrix[1]. A simple signal in the direction $\mathbf{n}$, as defined by equation (5.1), is represented by the tensor

$$\mathbf{T} = A\hat{\mathbf{n}}\hat{\mathbf{n}}^T, \tag{5.2}$$

where A is some constant that may encode other information than orientation, such as certainty or local signal energy. It is obvious that this representation maps $\hat{\mathbf{n}}$ and $-\hat{\mathbf{n}}$ to the same tensor and that the orientation can be recovered from the eigensystem of $\mathbf{T}$.

By design the orientation tensor satisfies the following two conditions:

[1] Symmetric matrices constitute a subclass of tensors. Readers who are more familiar with matrix algebra than with tensor algebra may safely substitute "symmetric matrix" for "tensor" throughout this chapter.

Invariance The normalized tensor $\hat{\mathbf{T}} = \frac{\mathbf{T}}{\|\mathbf{T}\|}$ does not depend on the function h in equation (5.1).

Equivariance The orientation tensor locally preserves the angle metric of the original space, i.e.

$$\|\delta\hat{\mathbf{T}}\| \propto \|\delta\hat{\mathbf{n}}\|. \tag{5.3}$$

The tensor norm used above is the Frobenius norm, $\|\mathbf{T}\|^2 = \mathrm{tr}\,(\mathbf{T}^T\mathbf{T})$.

5.2.2 Estimation

The orientation tensor can be computed by means of the responses of a set of quadrature filters. Each quadrature filter is spherically separable and real in the Fourier domain,

$$F_k(\mathbf{u}) = R(\|\mathbf{u}\|)D_k(\hat{\mathbf{u}}), \tag{5.4}$$

where the radial function R can be chosen more or less arbitrary, with typical design restrictions given by desired center frequency, bandwidth, locality, and scale. The directional function is given by

$$D_k(\hat{\mathbf{u}}) = \begin{cases} (\hat{\mathbf{u}}^T\hat{\mathbf{n}}_k)^2, & \hat{\mathbf{u}}^T\hat{\mathbf{n}}_k > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{5.5}$$

where $\{\hat{\mathbf{n}}_k\}$ is a set of direction vectors, usually evenly distributed in the signal space. It turns out that the minimum number of filters is 3 in 2D, 6 in 3D, and 12 in 4D.

The orientation tensor is constructed from the magnitudes of the filter responses $\{q_k\}$ at each point by

$$\mathbf{T} = \sum_k |q_k|\,\mathbf{M}_k, \tag{5.6}$$

where $\{\mathbf{M}_k\}$ are the duals of the outer product tensors $\{\hat{\mathbf{n}}_k\hat{\mathbf{n}}_k^T\}$.

5.2.3 Interpretation for Non-Simple Signals

The above construction is guaranteed to give a tensor as defined in equation (5.2) only for simple signals. For non-simple signals the tensor is analyzed by means of the eigenvalue decomposition, which can be written as

$$\mathbf{T} = \sum_k \lambda_k\,\hat{\mathbf{e}}_k\hat{\mathbf{e}}_k^T, \tag{5.7}$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$ are the eigenvalues and $\{\hat{\mathbf{e}}_k\}$ are the corresponding eigenvectors. In 3D, e.g., this can be rewritten as

$$\mathbf{T} = (\lambda_1 - \lambda_2)\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T + (\lambda_2 - \lambda_3)(\hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T + \hat{\mathbf{e}}_2\hat{\mathbf{e}}_2^T) + \lambda_3\mathbf{I}. \tag{5.8}$$

The tensor is here represented as a linear combination of three tensors. The first corresponds to a simple neighborhood, i.e. locally planar, the second to a rank 2

neighborhood, i.e. locally constant on lines, and the last term corresponds to an isotropic neighborhood, e.g. non-directed noise. For further details the reader is referred to [40].

5.3 Orientation Functionals

Here we take the view that the orientation tensor is an instance of a new concept called orientation functionals, defined below.

Let $\mathcal{U}$ denote the set of unit vectors in $\mathcal{R}^N$,

$$\mathcal{U} = \{\mathbf{u} \in \mathcal{R}^N; \|\mathbf{u}\| = 1\}. \tag{5.9}$$

An orientation functional ϕ is a mapping

$$\phi : \mathcal{U} \longrightarrow \mathcal{R}_+ \cup \{0\} \tag{5.10}$$

that to each direction vector assigns a non-negative real value. The value is interpreted as a measure of how well the signal locally is consistent with an orientation hypothesis in the given direction. Since we do not distinguish between two opposite directions, we require that ϕ be even, i.e. that

$$\phi(-\mathbf{u}) = \phi(\mathbf{u}), \qquad \text{all } \mathbf{u} \in \mathcal{U}. \tag{5.11}$$

We also set some restrictions on the mapping from signal neighborhoods to the associated orientation functionals. The signal f is assumed to be expressed in a local coordinate system, so that we always discuss the local orientation at the origin.

1. Assume that the signal is rotated around the origin, so that

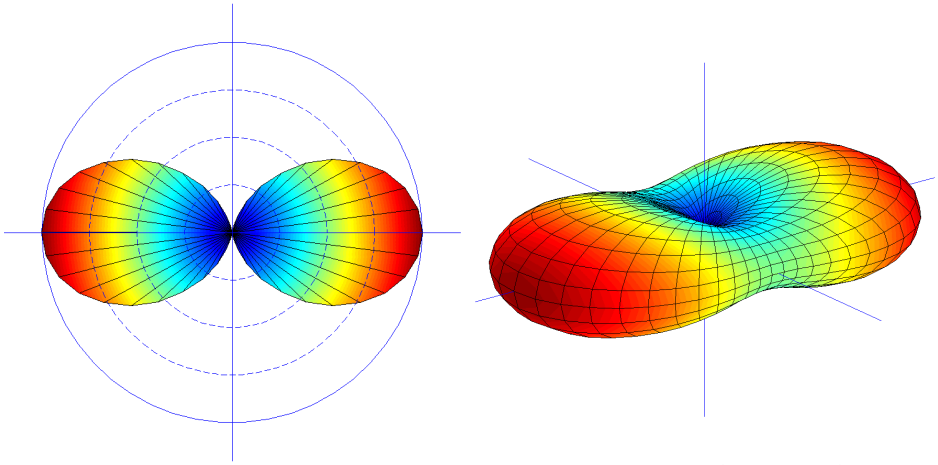
   $f(\mathbf{x})$ is replaced by $\tilde{f}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$, where $\mathbf{R}$ is a rotation matrix. Then the orientation functional $\tilde{\phi}$ associated to $\tilde{f}$ should relate to ϕ by $\tilde{\phi}(\mathbf{u}) = \phi(\mathbf{R}\mathbf{u})$, i.e. be rotated in the same way. This relation should also hold for other orthogonal matrices $\mathbf{R}$, characterized by $\mathbf{R}^T\mathbf{R} = \mathbf{I}$. These matrices represent isometric transformations, which in addition to rotations also include reflections and combinations of rotation and reflection.

2. In directions along which the signal is constant, ϕ should be zero.

3. For a simple signal in the direction $\hat{\mathbf{n}}$, ϕ should have its maximum value for $\hat{\mathbf{n}}$ and $-\hat{\mathbf{n}}$. It should also decrease monotonically as the angle to the closer of these two directions increases.

4. If a constant is added to the signal, ϕ should not change, i.e. the orientation functional should be invariant to the DC level.

5. If the signal is multiplied by a positive constant α, $\tilde{f}(\mathbf{x}) = \alpha f(\mathbf{x})$, the new orientation functional should be proportional to the old one, $\tilde{\phi}(\mathbf{u}) = \beta\phi(\mathbf{u})$, where the positive constant β is not necessarily equal to α but should vary monotonically with α.



(a) $\phi(\mathbf{u}) = \mathbf{u}^T \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \mathbf{u}$ (b) $\phi(\mathbf{u}) = \mathbf{u}^T \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{u}$

Figure 5.1: Polar plots of orientation functionals.

6. If the signal values are negated, ϕ should remain unchanged.

One might also think that the orientation functional should remain unchanged if the signal is scaled. This is, however, not the case. Orientation is a local property and a signal may look completely different at different scales. Thus we have the following non-requirement.

7. If the signal is uniformly scaled, $\tilde{f}(\mathbf{x}) = f(\lambda\mathbf{x})$, $|\lambda| \neq 1$, no additional restriction is set on the behavior of ϕ.

To transform an orientation tensor into an orientation functional, we simply use the construction

$$\phi_{\mathbf{T}}(\mathbf{u}) = \mathbf{u}^T \mathbf{T} \mathbf{u}. \tag{5.12}$$

Hence the orientation tensors are the subclass of orientation functionals which are positive semidefinite[2] quadratic forms in $\mathbf{u}$.

Orientation functionals can in 2D and 3D be illustrated by polar plots, as shown in figure 5.1. For a generalization of the orientation functional concept, see section 5.9.

[2]As it happens, the estimation method described in section 5.2.2 can sometimes yield an indefinite tensor. We will not consider that case further.

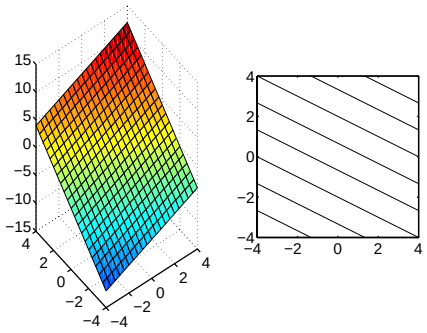


Figure 5.2: Linear neighborhood, $f(x, y) = x + 2y$.

5.4 Tensor Estimation Based on Polynomial Expansion

To estimate orientation we use the assumption that local projection onto second degree polynomials gives sufficient information. In other words we perform a polynomial expansion to obtain the coefficients $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$ of the local signal model

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c, \tag{5.13}$$

as described in the previous chapter.

To determine how the orientation tensor should be constructed from the coefficients of the local signal model, we start by studying purely linear and quadratic neighborhoods.

5.4.1 Linear Neighborhoods

A linear neighborhood can always be written as

$$f(\mathbf{x}) = \mathbf{b}^T \mathbf{x} \tag{5.14}$$

for some vector $\mathbf{b}$. Obviously this implies that the signal is simple with orientation given by $\mathbf{b}$. It should be clear that we get a suitable orientation tensor from the construction

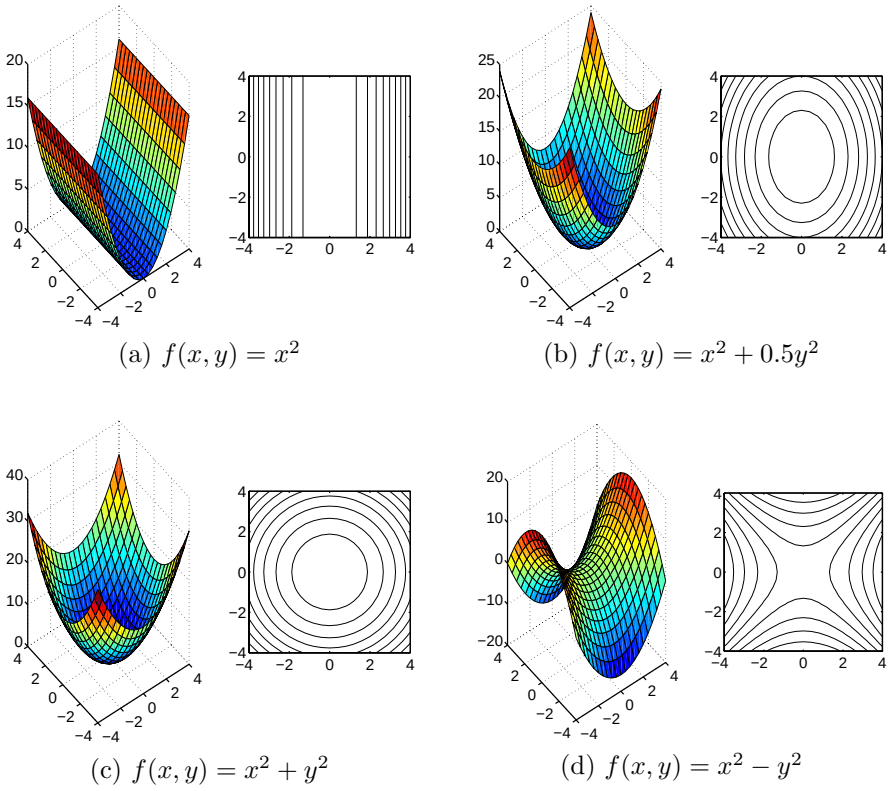
$$\mathbf{T} = \mathbf{b}\mathbf{b}^T. \tag{5.15}$$

An illustration of a linear neighborhood in 2D is given in figure 5.2.

5.4.2 Quadratic Neighborhoods

For quadratic neighborhoods,

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x}, \tag{5.16}$$



(a) $f(x, y) = x^2$ (b) $f(x, y) = x^2 + 0.5y^2$

(c) $f(x, y) = x^2 + y^2$ (d) $f(x, y) = x^2 - y^2$

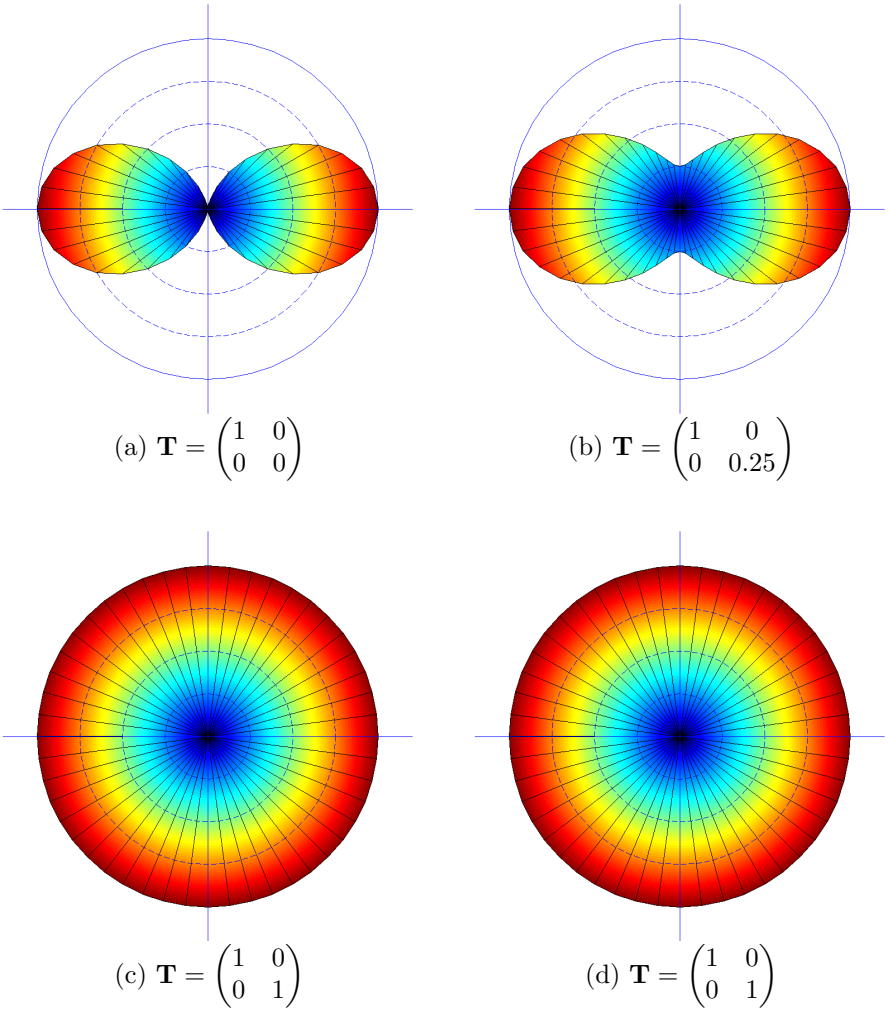
Figure 5.3: Quadratic neighborhoods.

the situation is more complicated. These neighborhoods are simple if and only if $\mathbf{A}$ is of rank 1. To get an idea of how to deal with higher rank neighborhoods we take a look at four different neighborhoods in 2D, depicted in figure 5.3. In (a) we have $f(x, y) = x^2$, a simple signal, so the orientation is clearly horizontal. In (b), where $f(x, y) = x^2 + 0.5y^2$, the horizontal direction still dominates but less distinctly. In (c) we have the perfectly isotropic neighborhood $f(x, y) = x^2 + y^2$, where no direction can be preferred. The signal illustrated in (d), $f(x, y) = x^2 - y^2$ is more confusing. Although it can be argued that it is constant on the two lines $y = \pm x$, this is not sufficient to consider it a simple signal in either direction. In fact we treat this signal too as completely isotropic, in a local orientation sense.

Analogously to the linear case we get a suitable orientation tensor by the construction

$$\mathbf{T} = \mathbf{A}\mathbf{A}^T. \tag{5.17}$$

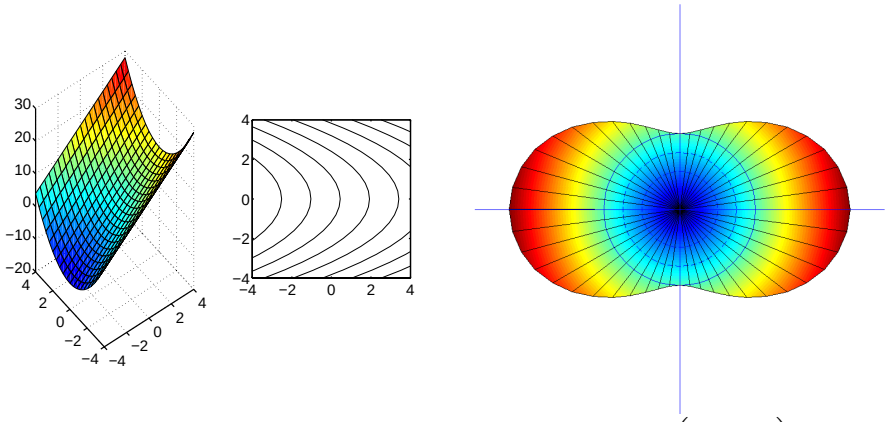
The tensors corresponding to the quadratic neighborhoods in figure 5.3 are given in figure 5.4 together with their polar plots.



(a) $\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ (b) $\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & 0.25 \end{pmatrix}$

(c) $\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ (d) $\mathbf{T} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

Figure 5.4: Tensors corresponding to quadratic neighborhoods.



(a) $f(x, y) = 3x + y^2$ (b) $\mathbf{T} = \begin{pmatrix} 2.25 & 0 \\ 0 & 1 \end{pmatrix}$

Figure 5.5: Linear plus quadratic neighborhood and corresponding tensor.

5.4.3 General Neighborhoods

For a general neighborhood we consider the local signal model

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c. \tag{5.18}$$

Here we add the tensors which would be obtained from the linear and quadratic components separately, i.e.

$$\mathbf{T} = \mathbf{A}\mathbf{A}^T + \gamma \mathbf{b}\mathbf{b}^T, \tag{5.19}$$

where γ is a non-negative weight factor. An example of a linear plus quadratic neighborhood is given in figure 5.5 together with the polar plot of the corresponding tensor for $\gamma = 0.25$. As we can see from the example, the proper value of γ depends on the scale at which we study the orientation. At a small scale the linear component should dominate while at a large scale the quadratic part is more significant. In general the value of γ should decrease when the size of the neighborhood under consideration becomes larger. Further discussion on the choice of γ can be found in sections 5.5 and 5.7.3.

5.5 Properties of the Estimated Tensor

Ideally we would like the estimated tensor to exactly satisfy the requirements of an orientation functional, listed in section 5.3. This is indeed the case if we restrict ourselves to the ideal case of continuous signals with constant certainty and require that the applicability be isotropic.

To begin with we can notice that from the construction of the tensor, $\mathbf{T} = \mathbf{A}\mathbf{A}^T + \gamma \mathbf{b}\mathbf{b}^T$, it is clear that $\mathbf{T}$ is symmetric and positive semidefinite. Thus the

corresponding functional $\phi_{\mathbf{T}}(\mathbf{u}) = \mathbf{u}^T\mathbf{T}\mathbf{u}$ maps to non-negative real values and is even because $\phi_{\mathbf{T}}(-\mathbf{u}) = (-\mathbf{u})^T\mathbf{T}(-\mathbf{u}) = \mathbf{u}^T\mathbf{T}\mathbf{u} = \phi_{\mathbf{T}}(\mathbf{u})$.

To verify the numbered requirements in section 5.3 we assume that we have a signal f, a projection onto a quadratic polynomial according to equation (5.13), the corresponding tensor $\mathbf{T}$ given by equation (5.19), and the corresponding orientation functional $\phi(\mathbf{u}) = \mathbf{u}^T\mathbf{T}\mathbf{u}$.

1. If f is rotated to yield $\tilde{f}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$, the projection onto a quadratic polynomial is rotated similarly,

$$\tilde{f}(\mathbf{x}) \sim (\mathbf{R}\mathbf{x})^T\mathbf{A}(\mathbf{R}\mathbf{x}) + \mathbf{b}^T\mathbf{R}\mathbf{x} + c = \mathbf{x}^T(\mathbf{R}^T\mathbf{A}\mathbf{R})\mathbf{x} + (\mathbf{R}^T\mathbf{b})^T\mathbf{x} + c. \quad (5.20)$$

   This follows from the fact that the set of quadratic polynomials is closed under rotation and the assumptions that the certainty is constant (and thus isotropic) and that the applicability is isotropic. Now we get the tensor corresponding to $\tilde{f}$ by

$$\begin{aligned}\tilde{\mathbf{T}} &= (\mathbf{R}^T\mathbf{A}\mathbf{R})(\mathbf{R}^T\mathbf{A}\mathbf{R})^T + \gamma(\mathbf{R}^T\mathbf{b})(\mathbf{R}^T\mathbf{b})^T \\ &= \mathbf{R}^T\mathbf{A}\mathbf{A}^T\mathbf{R} + \gamma\mathbf{R}^T\mathbf{b}\mathbf{b}^T\mathbf{R} = \mathbf{R}^T\mathbf{T}\mathbf{R}.\end{aligned} \quad (5.21)$$

   From this it follows that

$$\tilde{\phi}(\mathbf{u}) = \mathbf{u}^T\tilde{\mathbf{T}}\mathbf{u} = \mathbf{u}^T\mathbf{R}^T\mathbf{T}\mathbf{R}\mathbf{u} = (\mathbf{R}\mathbf{u})^T\mathbf{T}(\mathbf{R}\mathbf{u}) = \phi(\mathbf{R}\mathbf{u}). \quad (5.22)$$

   The only property of $\mathbf{R}$ that we have used above is $\mathbf{R}\mathbf{R}^T = \mathbf{I}$, so this derivation is equally valid for other isometric transformations.

2. Assume that f is constant along the first coordinate axis, and let $\mathbf{u}_1$ be the corresponding direction vector. Then f does not depend on the first variable and neither does the projection onto a quadratic polynomial. Thus we have $\mathbf{A}\mathbf{u}_1 = \mathbf{A}^T\mathbf{u}_1 = \mathbf{0}$ and $\mathbf{b}^T\mathbf{u}_1 = 0$ so that

$$\phi(\mathbf{u}_1) = \mathbf{u}_1^T\mathbf{A}\mathbf{A}^T\mathbf{u}_1 + \gamma\mathbf{u}_1^T\mathbf{b}\mathbf{b}^T\mathbf{u}_1 = 0. \quad (5.23)$$

   If f is constant along some other direction the conclusion still follows because property 1 allows us to rotate this direction onto $\mathbf{u}_1$.

3. If f is N-dimensional and simple in the direction $\hat{\mathbf{n}}$, there is a set of $N-1$ orthogonal directions along which it is constant. From property 2 it follows that these directions are eigenvectors of $\mathbf{T}$ corresponding to the eigenvalue zero and as a consequence $\mathbf{T}$ is at most of rank one. Hence we have

$$\mathbf{T} = \alpha\hat{\mathbf{n}}\hat{\mathbf{n}}^T, \quad (5.24)$$

   for some non-negative α and

$$\phi(\mathbf{u}) = \alpha\mathbf{u}^T\hat{\mathbf{n}}\hat{\mathbf{n}}^T\mathbf{u} = \alpha(\hat{\mathbf{n}}^T\mathbf{u})^2 = \alpha\cos^2\theta, \quad (5.25)$$

   where θ is the angle between $\hat{\mathbf{n}}$ and $\mathbf{u}$.

4. If a constant is added to $\mathbf{f}$, this only affects the value of c in the projection onto a quadratic polynomial. Since c is discarded in the construction of $\mathbf{T}$, the tensor remains unchanged.

5. If the amplitude of the signal is multiplied by a constant α, $\tilde{f}(\mathbf{x}) = \alpha f(\mathbf{x})$, the projection is multiplied by the same constant, i.e.

$$\tilde{f}(\mathbf{x}) \sim \mathbf{x}^T(\alpha\mathbf{A})\mathbf{x} + (\alpha\mathbf{b})^T\mathbf{x} + \alpha c. \tag{5.26}$$

   Hence the new tensor is given by

$$\tilde{\mathbf{T}} = (\alpha\mathbf{A})(\alpha\mathbf{A})^T + \gamma(\alpha\mathbf{b})(\alpha\mathbf{b})^T = \alpha^2\mathbf{T} \tag{5.27}$$

   and the corresponding orientation functional becomes

$$\tilde{\phi}(\mathbf{u}) = \alpha^2\phi(\mathbf{u}). \tag{5.28}$$

6. If the signal values are negated, the tensor is unchanged. This follows from equation (5.27) with $\alpha = -1$.

7. If the signal is uniformly scaled, $\tilde{f}(\mathbf{x}) = f(\lambda\mathbf{x})$, things become more complicated. To begin with, if the signal is a quadratic polynomial, we have

$$\tilde{f}(\mathbf{x}) = (\lambda\mathbf{x})^T\mathbf{A}(\lambda\mathbf{x}) + \mathbf{b}^T(\lambda\mathbf{x}) + c = \mathbf{x}^T(\lambda^2\mathbf{A})\mathbf{x} + (\lambda\mathbf{b})^T\mathbf{x} + c, \tag{5.29}$$

   and the new orientation tensor is given by

$$\tilde{\mathbf{T}} = \lambda^4\mathbf{A}\mathbf{A}^T + \lambda^2\gamma\mathbf{b}\mathbf{b}^T. \tag{5.30}$$

   Hence the relative weight between the quadratic and linear parts of the tensor is altered. For a general signal the projection onto a polynomial may change arbitrarily, because it may look completely different at varying scales. In the case where the applicability is scaled identically with the signal, however, the projection is scaled according to equation (5.29) and to get a new tensor proportional to the old one, we need to scale the weight factor γ by λ^2.

In practice, with discretized signals of limited extent, we cannot guarantee that all of these requirements be perfectly fulfilled. The primary reason is that we cannot even perform an arbitrary rotation of a discretized signal without introducing errors, so we cannot really hope to do any better with the orientation descriptor.

5.6 Computational Complexity

The major part of the computational complexity of the tensor estimation algorithm is the computation of the polynomial expansion. Thus we use the same notation as in section 4.4 with d dimensionality, n size of applicability, and NC, C, SC, and SNC being names of the different methods to compute polynomial expansion.

Since we do not use the local DC level in the tensor construction, we do not have to compute the expansion coefficient corresponding to the constant basis

function. This gives a reduction by one filter for the C method and by $d+1$ multiplications per point for the SC method.

The tensor construction from the expansion coefficients requires $\frac{d(d+1)(d+2)}{2}$ multiplications per point.

The memory overhead differs slightly from the polynomial expansion case. The $m = \frac{(d+1)(d+2)}{2}$ expansion coefficients, which in the polynomial expansion algorithm were the output, are now only of temporary interest. Since the number of independent tensor elements, $\frac{d(d+1)}{2}$, is smaller, the difference must count as memory overhead. This is not a problem for the NC method, however, since we can make the computation all the way to the tensor elements point for point.

All this is summarized in tables 5.1 and 5.2, which are direct counterparts to tables 4.1 and 4.2 for the complexity of the polynomial expansion only.

The discussion about the relative merits of the different methods in section 4.4 is valid also in the context of tensor estimation.

5.7 Evaluation

The tensor estimation algorithm has been evaluated on a 3D test volume consisting of concentric spherical shells. The volume is $64 \times 64 \times 64$ and selected slices are displayed in figure 5.6. Except at the center of the volume the signal is locally planar and all possible orientations are present. As in [6, 63] the performance of the tensors is measured by an angular RMS error

$$\Delta\phi = \arcsin\left(\sqrt{\frac{1}{2L}\sum_{l=1}^{L}\|\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T\|^2}\right), \tag{5.31}$$

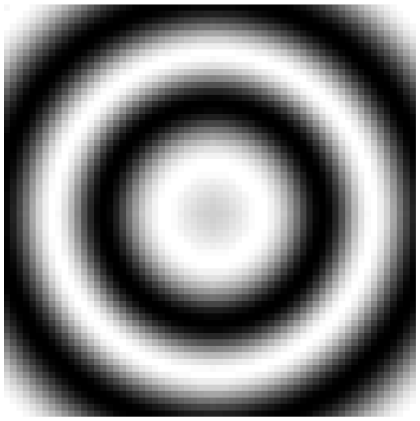
where $\hat{\mathbf{x}}$ is a unit vector in the correct orientation, $\hat{\mathbf{e}}_1$ is the eigenvector corresponding to the largest eigenvalue of the estimated tensor, and L is the number of points. To avoid border effects and irregularities at the center of the volume, the sum is only computed for points at a radius between 0.16 and 0.84, with respect to normalized coordinates. As is shown in appendix F, the angular RMS error can

Method	Time complexity	Memory overhead
NC	$\frac{d^4}{24}n^d$	0
C	$\frac{d^2}{2}n^d$	d
SC	$\frac{d^3}{6}n$	$d+1$
SNC	$\frac{d^5}{120}n + \frac{d^6}{48}$	$\frac{d^4}{24}$

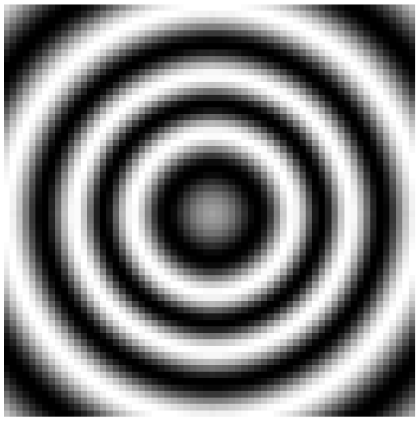
Table 5.1: Asymptotic complexities, d and n large, leading terms.

	Time complexity			Memory overhead		
Method	$d = 2$	$d = 3$	$d = 4$	$d = 2$	$d = 3$	$d = 4$
NC	$21n^2 + 104$	$45n^3 + 350$	$85n^4 + 965$	0	0	0
C	$5n^2 + 12$	$9n^3 + 30$	$14n^4 + 60$	2	3	4
SC	$9n + 19$	$19n + 42$	$34n + 78$	3	4	5
SNC	$29n + 104$	$74n + 350$	$159n + 965$	18	39	75

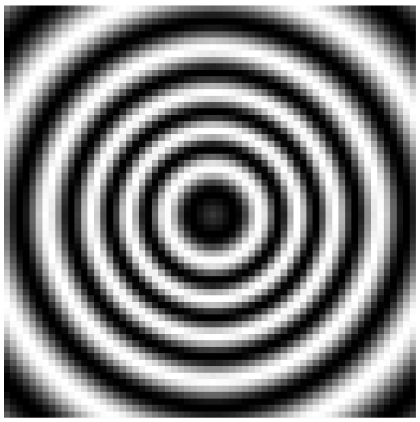
Table 5.2: Time complexity and memory overhead for 2D, 3D, and 4D.



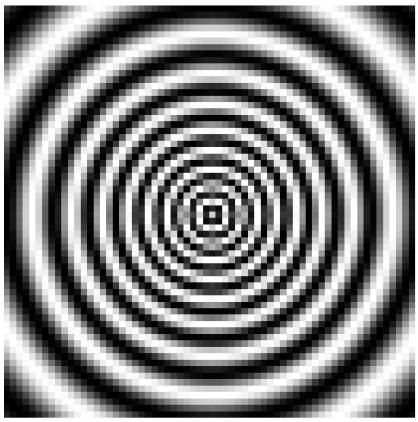
(a) slice 5



(b) slice 14

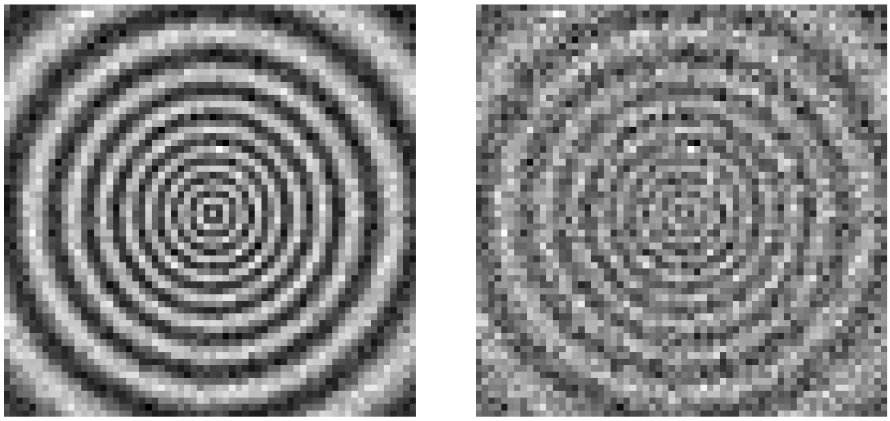


(c) slice 21



(d) slice 32

Figure 5.6: Slices from the 64-cube test volume.



(a) SNR = 10 dB (b) SNR = 0 dB

Figure 5.7: White noise added to slice 32 of the test volume.

equivalently be written as

$$\Delta\phi = \arccos\left(\sqrt{\frac{1}{L}\sum_{l=1}^{L}(\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2}\right). \tag{5.32}$$

Although the slices in figure 5.6 may give the impression that the volume contains structures at a wide range of scales, this is not the case from a 3D perspective. As can be seen from slice 32, the distance between two shells varies between about 3 and 6 pixels within the tested part of the volume. Hence it is possible to obtain very good performance by orientation estimation at a single scale.

The algorithm has also been tested on degraded versions of the test volume, where white noise has been added to get a signal to noise ratio of 10 dB and 0 dB respectively. One slice of each of these are shown in figure 5.7.

5.7.1 The Importance of Isotropy

As we saw in section 5.5, isotropy is a theoretically important property of the applicability. To test this in practice a number of different applicabilities have been evaluated. The test set consists of:

- Cubes of four different sizes, with sides being 3, 5, 7, and 9 pixels wide.
- A sphere of radius 3.5 pixels.
- The same sphere but oversampled, i.e. sampled regularly at 10×10 points per pixel and then averaged. The result is a removal of jaggies at the edges and a more isotropic applicability.
- A 3D cone of radius 4 pixels.

shape	∞	10 dB	0 dB
cube $3 \times 3 \times 3$	3.74°	7.27°	24.06°
cube $5 \times 5 \times 5$	13.50°	14.16°	18.48°
cube $7 \times 7 \times 7$	22.99°	23.57°	27.30°
cube $9 \times 9 \times 9$	30.22°	30.64°	33.62°
Sphere	6.69°	8.20°	15.34°
Sphere, oversampled	0.85°	5.78°	14.30°
Cone	1.39°	6.10°	13.89°
Cone, oversampled	0.28°	5.89°	14.13°
Tent, oversampled	21.38°	21.86°	25.16°
Binomial $5 \times 5 \times 5$	2.07°	3.74°	10.76°
Binomial $7 \times 7 \times 7$	2.00°	4.46°	11.42°
Binomial $9 \times 9 \times 9$	1.97°	6.68°	13.65°
Gaussian, $\sigma = 1.2$	0.17°	3.53°	10.88°

Table 5.3: Evaluation of different applicabilities.

- The same cone oversampled.
- A "tent" shape, 8 pixels wide, oversampled.
- Binomial applicabilities with sides being 5, 7, and 9 pixels wide.
- A Gaussian with standard deviation 1.2, sampled at $9 \times 9 \times 9$ points.

The first twelve applicabilities are illustrated in figure 5.8 in form of their 2D counterparts. The Gaussian can be found in figure 5.9 (b).

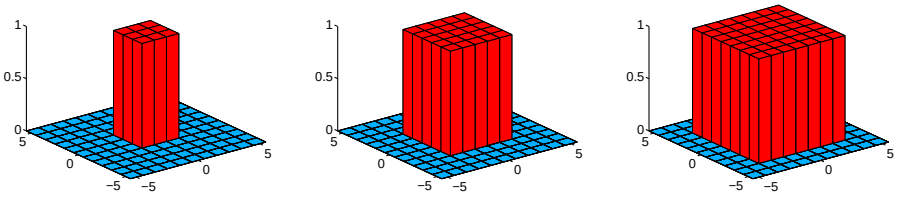
The results are listed in table 5.3 and we can see that the cube and tent shapes, which are highly anisotropic,[3] perform significantly worse than the more isotropic shapes. This is of particular interest since the cube applicability corresponds to the naive use of an unweighted subspace projection; cf. the cubic facet model, discussed in sections 3.9.2 and 4.8.2.

The main reason why the cubes are anisotropic is that they extend farther into the corners than along the axes. The spheres and the cones eliminate this phenomenon by being cut off at some radius. Still there is a marked improvement in isotropy when these shapes are oversampled, which can clearly be seen in the results from the noise-free volume.

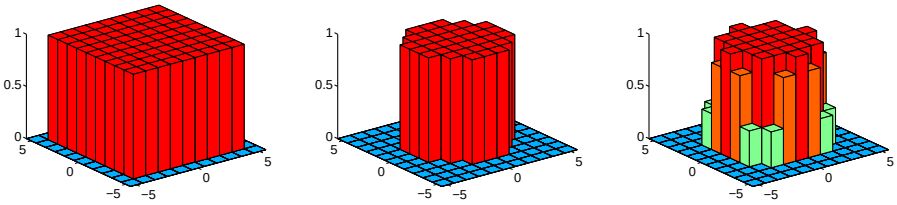
The difference between the spheres and the cones is that the latter have a gradial decline in the importance given to points farther away from the center. We can see that this makes a difference, primarily when there is no noise, but that the significance of isotropy is much larger can clearly be seen from the poor results of the tent shape.

The Gaussian, finally, turns out to yield superior performance, which is very fortunate considering that this shape is separable and therefore allows computation of the polynomial expansion with the fast algorithm described in section 4.3.2. The

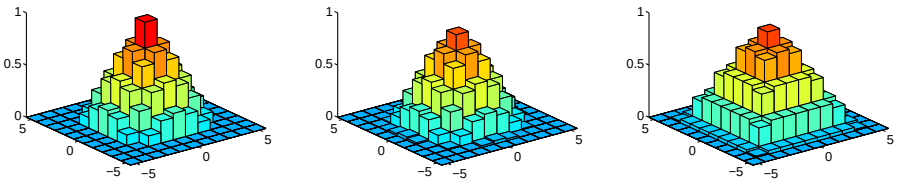
[3] The $3 \times 3 \times 3$ cube is actually too small to be significantly anisotropic.



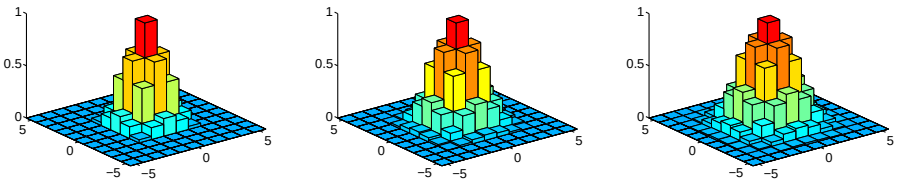
(a) Cube $3 \times 3 \times 3$ (b) Cube $5 \times 5 \times 5$ (c) Cube $7 \times 7 \times 7$



(d) Cube $9 \times 9 \times 9$ (e) Sphere, radius 3.5 (f) Sphere, oversampled

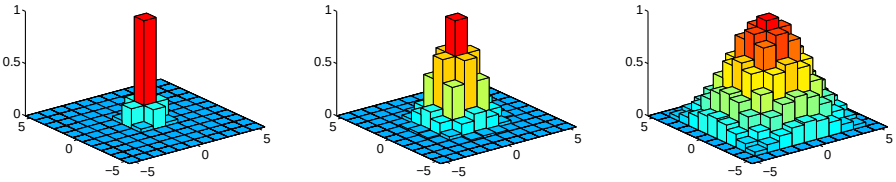


(g) Cone, radius 4 (h) Cone, oversampled (i) Tent, oversampled



(j) Binomial $5 \times 5 \times 5$ (k) Binomial $7 \times 7 \times 7$ (l) Binomial $9 \times 9 \times 9$

Figure 5.8: Applicabilities used to test orientation estimation.



(a) $\sigma = 0.3$ (b) $\sigma = 1.2$ (c) $\sigma = 4.8$

Figure 5.9: Gaussian applicabilities with different standard deviations.

binomials are also separable but not quite as isotropic and perform clearly worse on the noise-free volume.

5.7.2 Gaussian Applicabilities

With Gaussian applicabilities there is only one parameter to vary, the standard deviation σ. The Gaussian must be truncated, however, and with the separable algorithm the truncation is implicitly made to a cube of some size. Figure 5.9 shows three Gaussians with widely varying standard deviations, truncated to a cube with side 9. There are three aspects to note with respect to the choice of σ:

1. The size of the applicability should match the scale of the structures we want to estimate orientation for.

2. For small applicabilities the estimation is typically more sensitive to noise than for larger ones.

3. If the standard deviation is large relative to the size to which the Gaussian is truncated, the contributions from the corners tend to make the applicability anisotropic, as is illustrated in figure 5.9(c). Fortunately the Gaussian decreases very fast sufficiently far from the origin, so with a proper choice of the truncation size the Gaussian remains very isotropic.

The results are shown in figure 5.10. It is noteworthy that the anisotropic tendencies affect the performance of the algorithm, in the absence of noise, quite significantly already for σ about 1.5. For very large σ the Gaussian approaches the cube applicability, which explains why the $11 \times 11 \times 11$ kernel ultimately performs worse than the $9 \times 9 \times 9$ kernel.

Section 4.5 discussed the effects of applying polynomial expansion to a lowpass filtered signal. If both the lowpass filter and the applicability are Gaussians, with standard deviations σ_1 and σ_2 respectively, this would in the continuous case be equivalent to a polynomial expansion of the original signal, where the applicability is Gaussian with standard deviation $\sqrt{\sigma_1^2 + \sigma_2^2}$. Figure 5.11 shows the angular errors when this approach, in the discrete case, is used in the tensor computations. For a given σ_2, σ_1 is chosen so that $\sigma_1^2 + \sigma_2^2 = 1.1^2$. The results would probably be much worse for small σ_2 if the test volume was more complex.

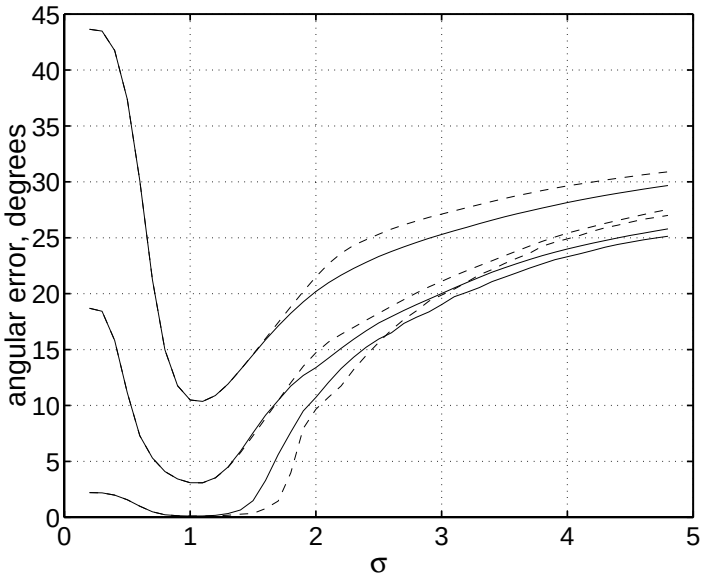


Figure 5.10: Angular errors for Gaussians with varying standard deviations. The solid lines refer to $9 \times 9 \times 9$ kernels while the dashed lines refer to $11 \times 11 \times 11$ kernels. The three curve pairs are for 0, 10, and ∞ SNR respectively.

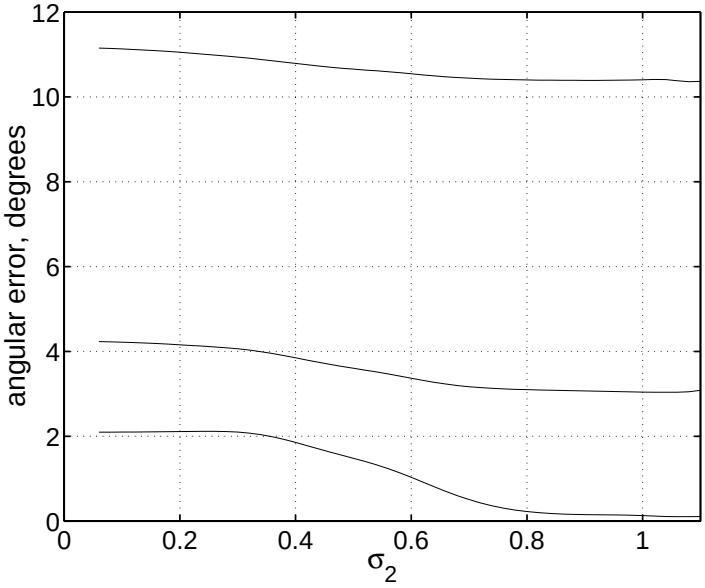
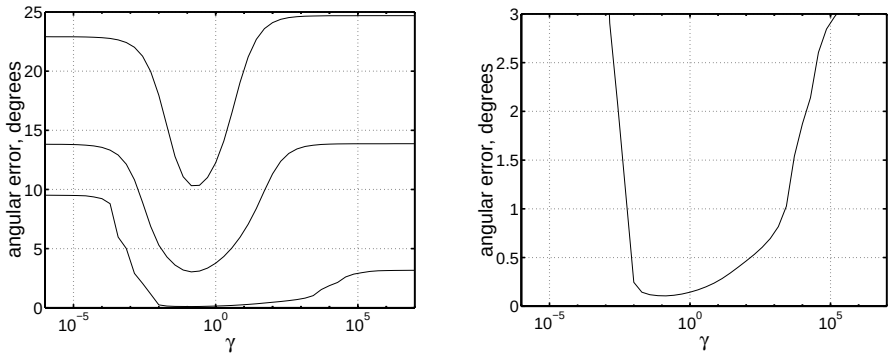


Figure 5.11: Angular errors for tensors computed on a lowpass filtered signal.



(a) (b)

Figure 5.12: (a) Angular errors for varying γ values. The three curves are for 0, 10, and ∞ SNR respectively. (b) A magnification of the results for the noise-free test volume.

We can see that the best result is obtained when no prefiltering is used. With a reduced σ_2 we can use a smaller applicability in the polynomial expansion, which reduces the computational complexity. It is not clear from this example how large reductions are safe, however.

5.7.3 Choosing γ

Another parameter in the tensor estimation algorithm is the relative weight for the linear and quadratic parts of the signal, γ. In the previous experiments γ has been chosen reasonably, with only a small optimization effort. To see how the value of γ typically affects the performance we have varied γ for a fixed Gaussian applicability with optimal standard deviation, 1.06. The results are shown in figure 5.12. We can clearly see that neither the linear nor the quadratic part are very good on their own but suitably weighted together they give much better results. We also observe that the linear part on its own works better than the quadratic part in the absence of noise, but that it is more noise sensitive. It is interesting to note here that the linear part, interpreted in terms of derivatives (see section 4.8.1), essentially is a gradient, which is a classic means to estimate orientation.

5.7.4 Best Results

Table 5.4 lists the best results obtained for different sizes of the applicability. All computations have been made with the separable algorithm and σ and γ have been tuned for each applicability size, n.

The results for $9 \times 9 \times 9$ applicabilities, and equivalently kernels of the same effective size, can readily be compared to the results given in [6, 63] for a sequential filter implementation of the quadrature filter based estimation algorithm described in section 5.2.2. As we can see in table 5.5, the algorithm proposed in this thesis

n	∞	10 dB	0 dB	kernel coefficients	total operations
3	0.87°	6.69°	23.18°	57	99
5	0.37°	3.51°	10.70°	85	127
7	0.15°	3.05°	10.26°	133	175
9	0.11°	3.03°	10.24°	171	213
11	0.11°	3.03°	10.24°	209	251

Table 5.4: Smallest angular errors for different kernel sizes.

SNR	Andersson, Wiklund, Knutsson	Farnebäck	Johansson	
	345 coeff.	171 coeff.	72 coeff.	90 coeff.
∞	0.76°	0.11°	0.78°	0.28°
10 dB	3.02°	3.03°	3.39°	3.05°
0 dB	9.35°	10.24°	10.25°	10.37°

Table 5.5: Comparison with Andersson, Wiklund & Knutsson [6, 63] and Johansson [54].

performs quite favorably[4] in the absence of noise while being somewhat more noise sensitive. Additionally it uses only half the number of kernel coefficients. Included in the table are also results for the approximative polynomial expansion algorithm by Johansson [54], described in section 4.6.1. This reduces the number of kernel coefficients further with only small impact on the angular errors.

5.8 Discussion

Although the orientation estimation algorithm has been shown to work very well, see also the results in section 6.6, there are still a number of areas where it could be improved.

5.8.1 Multiple Scales

The algorithm is quite selective with respect to the scale of the structures in the signal, which depending on the application may be either an advantage or a disadvantage. If it is necessary to estimate orientation over a large range of scales, the best solution probably is to compute a number of tensors at distinct scales and subsequently combine them into a single tensor. Preliminary experiments indicate

[4]To be fair it should be mentioned that the filters used in [6, 63] are claimed not to have been tuned at all for performance on the test volume. One would still guess that the available parameters have been chosen quite reasonably though.

that it is sufficient to simply add the tensors together, taking the scaling relations of equation (5.30) into account.

5.8.2 Different Radial Functions

The basis functions can in 2D be written in polar coordinates as

$$\{1, \rho\cos\phi, \rho\sin\phi, \rho^2\cos^2\phi, \rho^2\sin^2\phi, \rho^2\cos\phi\sin\phi\}. \tag{5.33}$$

It is easy to show that it is only the angular functions that are essential for the rotation equivariance and other properties of the tensor. The radial functions may well be something other than ρ for the linear basis functions and ρ^2 for the quadratic ones. One possibility would be to use the radial function ρ for all the basis functions except the constant. As a consequence both parts of the tensor would scale equally when both signal and applicability are scaled and there would be no need to adjust γ, cf. equation (5.30). Another possibility would be to try to obtain matching radial functions in the Fourier domain, which currently is not the case.

One should be aware, however, that changing the radial functions would destroy the separability of the basis functions.

5.8.3 Additional Basis Functions

It would be conceivable to expand the signal model, equation (5.13), e.g. with higher degree polynomials. It is not obvious that this would actually improve anything, however, but it would certainly increase the computational complexity.

To make the increased complexity worthwhile it would probably be necessary to find a way to ensure that the additional basis functions reduce the noise sensitivity of the algorithm, possibly by introducing some kind of redundancy.

5.9 Phase Functionals

This chapter concludes with a preliminary discussion about phase functionals, a possible generalization of orientation functionals to also include phase information. This gives us a novel and powerful representation for phase, or rather a combined orientation and phase representation.

With $\mathcal{U}$ defined by equation (5.9), a phase functional is a mapping

$$\theta : \mathcal{U} \longrightarrow \mathcal{C} \tag{5.34}$$

that to each direction vector assigns a complex value. The magnitude of the value is interpreted as a measure of the signal variation along the direction, while the argument of the value is interpreted as the local phase of the signal with respect to the direction. If we reverse the direction the magnitude should be unchanged while the argument should be negated. Hence we require that θ be Hermitian, i.e. that

$$\theta(-\mathbf{u}) = \theta(\mathbf{u})^*, \qquad \text{all } \mathbf{u} \in \mathcal{U}. \tag{5.35}$$

$\arg\phi$	$e^{i\arg\phi}$	interpretation
0	1	even, local maximum
$\frac{\pi}{2}$	i	odd, decreasing
π	-1	even, local minimum
$-\frac{\pi}{2}$	$-i$	odd, increasing

Table 5.6: Interpretation of local phase.

It is clear that by taking the magnitude[5] of a phase functional we obtain an orientation functional.

The method for estimation of orientation tensors in section 5.4 can be extended to estimation of phase after some preparations. To begin with we use a more liberal definition of signal phase than what is usually used. Instead of relating it to the phase of a sinusoidal it is interpreted as some relation between the odd and even parts of the signal, DC component excluded. A similar approach to the definition of phase is taken by Nordberg [74]. Table 5.6 lists the primary characteristics of the phase. With the signal model given by equation (5.13) it is clear that $\mathbf{A}$ represents the even part of the signal, excluding the DC component, while $\mathbf{b}$ represents the odd part. Thus it should be possible to construct a phase functional from $\mathbf{A}$ and $\mathbf{b}$.

Unfortunately we cannot use a quadratic form to represent phase, as we did with the tensor for orientation. The reason for this is that quadratic forms by necessity give even functionals, a property that is compatible with being Hermitian only if they are also real-valued, which would be useless in this context. A way to get around this is to add one dimension to the representation and use a quadratic form with respect to

$$\tilde{\mathbf{u}} = \begin{pmatrix} \mathbf{u} \\ 1 \end{pmatrix}. \tag{5.36}$$

By setting the phase tensor

$$\mathbf{P} = -\begin{pmatrix} \mathbf{A} & i\gamma\mathbf{b} \\ i\gamma\mathbf{b}^T & 0 \end{pmatrix}, \tag{5.37}$$

where γ has a similar role to that in the construction of the orientation tensor, we obtain the phase functional

$$\theta_{\mathbf{P}}(\mathbf{u}) = \tilde{\mathbf{u}}^T\mathbf{P}\tilde{\mathbf{u}} = -\mathbf{u}^T\mathbf{A}\mathbf{u} - i2\gamma\mathbf{b}^T\mathbf{u}. \tag{5.38}$$

If we take the magnitude of this phase functional we obtain an orientation functional that is different from the orientation tensor in section 5.4 but it is interesting to notice that the latter appears in

$$\tilde{\mathbf{u}}^T\mathbf{P}\mathbf{P}^*\tilde{\mathbf{u}} = \mathbf{u}^T(\mathbf{A}\mathbf{A}^T + \gamma^2\mathbf{b}\mathbf{b}^T + \gamma^2(\mathbf{b}^T\mathbf{b})\mathbf{I})\mathbf{u}, \tag{5.39}$$

with only an extra isotropic term in the tensor.

[5] Or the squared magnitude or something similar.

Chapter 6

Velocity Estimation

6.1 Introduction

If an image sequence is considered as a spatiotemporal volume, it is possible to use the orientation information in the volume for estimation of the motion in the sequence. In particular the tensor representation of orientation allows straightforward estimation of the motion, see e.g. [40, 49, 50, 51, 52, 53, 94] and section 6.2. The tensor can also be used more indirectly to provide constraints on the motion in order to estimate parameterized motion models, which is the basis for the methods developed in this chapter. Related approaches have also been used by Karlholm [61]. For overviews of other methods for motion estimation, the reader is referred to [8] and [19].

The algorithms presented in sections 6.3 and 6.4 have their origins in my master's thesis [21, 22], at that time using orientation tensors estimated by quadrature filters and with emphasis on segmentation of the motion field rather than on velocity estimation. The simplified algorithm in section 6.5 and the results in section 6.6 were first published in my licentiate thesis [23] and have subsequently also appeared in [2, 25, 28]. All these algorithms are novel and together with orientation tensors estimated by the algorithms from chapter 5 they give excellent performance, as demonstrated in section 6.6.

6.2 From Orientation to Motion

By stacking the frames of an image sequence onto each other we obtain a spatiotemporal image volume with two spatial dimensions and a third temporal dimension. It is easy to see that there is a strong correspondence between the motions in the image sequence and the orientations in the image volume. A moving line, e.g., is converted into a plane in the volume and from the orientation of the plane we can recover the velocity component perpendicular to the line. The fact that we can only obtain the perpendicular component is a fundamental limitation known as the aperture problem; the parallel velocity component of a linear structure cannot

be determined simply because it does not induce any change in the local signal. A moving point, on the other hand, is converted into a line in the volume and from the direction of the line we can obtain the true motion.

In terms of orientation tensors, the first case corresponds to a rank 1 tensor, where the largest eigenvector gives the orientation of the planar structure, while the second case corresponds to a rank 2 tensor, where the smallest eigenvector gives the direction along the linear structure. More precisely, with the tensor $\mathbf{T}$ expressed by the eigenvalue decomposition as in equation (5.7),

$$\mathbf{T} = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T + \lambda_3 \mathbf{e}_3 \mathbf{e}_3^T, \tag{6.1}$$

the velocity in the two cases can be computed by, taken from [40],

$$\begin{cases} \mathbf{v}_{\text{normal}} &= -x_3(x_1\hat{\boldsymbol{\xi}}_1 + x_2\hat{\boldsymbol{\xi}}_2)/(x_1^2 + x_2^2) \\ x_1 &= \hat{\mathbf{e}}_1^T\hat{\boldsymbol{\xi}}_1 \\ x_2 &= \hat{\mathbf{e}}_1^T\hat{\boldsymbol{\xi}}_2 \\ x_3 &= \hat{\mathbf{e}}_1^T\hat{\mathbf{t}} \end{cases} \quad \text{moving line case,} \tag{6.2}$$

and

$$\begin{cases} \mathbf{v} &= (x_1\hat{\boldsymbol{\xi}}_1 + x_2\hat{\boldsymbol{\xi}}_2)/x_3 \\ x_1 &= \hat{\mathbf{e}}_3^T\hat{\boldsymbol{\xi}}_1 \\ x_2 &= \hat{\mathbf{e}}_3^T\hat{\boldsymbol{\xi}}_2 \\ x_3 &= \hat{\mathbf{e}}_3^T\hat{\mathbf{t}} \end{cases} \quad \text{moving point case,} \tag{6.3}$$

where $\hat{\boldsymbol{\xi}}_1$ and $\hat{\boldsymbol{\xi}}_2$ are the orthogonal unit vectors defining the image plane and $\hat{\mathbf{t}}$ is a unit vector in the time direction.

One problem with this approach to velocity estimation is that we at each point must decide whether we can compute true velocity or have to be content with the normal component. Another problem is robustness. The method is sensitive both to noise and to errors in the tensor estimation. A common method to increase the robustness is averaging of the tensors in a neighborhood of each point, discussed further in section 6.5.

6.3 Estimating a Parameterized Velocity Field

Rather than estimating the velocity from the tensors point for point we assume that the velocity field over a region can be parameterized according to some motion model and we use all the tensors in the region to compute the parameters. To begin with we assume that we somehow have access to a region in the current frame of the sequence, within which the motion can be described, at least approximately, by some relatively simple parametric motion model.

6.3.1 Motion Models

The simplest possible motion model is to assume that the velocity is constant over the region,

$$\begin{aligned} v_x(x,y) &= a, \\ v_y(x,y) &= b, \end{aligned} \tag{6.4}$$

where x and y are the spatial coordinates, v_x and v_y are the x and y components of the velocity, and a and b are the model parameters. Geometrically this motion model corresponds to objects undergoing a pure translation under orthographic projection. A more powerful alternative is the affine motion model,

$$\begin{aligned} v_x(x,y) &= ax + by + c, \\ v_y(x,y) &= dx + ey + f, \end{aligned} \tag{6.5}$$

which applies to planar patches undergoing rigid body motion, i.e. translation plus rotation, under orthographic projection. To also account for a perspective projection we need the eight parameter motion model,

$$\begin{aligned} v_x(x,y) &= a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy, \\ v_y(x,y) &= a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2. \end{aligned} \tag{6.6}$$

The usefulness of these models does of course depend on the application but it is useful to notice that sufficiently far away most surfaces can be approximated as planar and if the distance to the scene is much larger than the variation in distance within the scene, perspective projection can be approximated by orthographic projection. More details on the derivation of these motion models can be found in [19].

Of course it would be possible to design other motion models for specific expected velocity fields, but we will only consider those listed above in this presentation. Requirements on the motion models in order to be useful with the methods developed in this chapter are given in section 6.3.3.

6.3.2 Cost Functions

A 2D velocity vector $(v_x, v_y)^T$, measured in pixels per frame, can be extended to a 3D spatiotemporal directional vector $\mathbf{v}$ and a unit directional vector $\hat{\mathbf{v}}$ by

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix}, \quad \hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}. \tag{6.7}$$

Ideally, in the case of a constant translation[1], we obtain a spatiotemporal neighborhood which is constant in the $\hat{\mathbf{v}}$ direction. By property 2 of section 5.3 we therefore have the constraint that

$$\phi_{\mathbf{T}}(\hat{\mathbf{v}}) = \hat{\mathbf{v}}^T \mathbf{T} \hat{\mathbf{v}} = 0. \tag{6.8}$$

[1] In the image plane.

This is consistent with the discussion in section 6.2 since in the moving point case, the least eigenvector has eigenvalue zero and is parallel to $\hat{\mathbf{v}}$, while in the moving line case, all $\hat{\mathbf{v}}$ with the correct normal velocity component satisfies equation (6.8).

In a less ideal case, where the motion is not a translation, where the motion is a translation but it varies over time, or in the presence of noise, we typically have to deal with rank 3 tensors, meaning that the constraint (6.8) cannot be fulfilled. As discussed in section 5.3, the interpretation of the orientation functional $\phi_{\mathbf{T}}$ is that the value in a given direction is a measure of how well the signal locally is consistent with an orientation hypothesis in that direction. In this case we are searching for directions along which the signal varies as little as possible and thus we wish to minimize $\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}}$.

Rewriting the tensor as

$$\begin{aligned}\mathbf{T} &= \lambda_1\mathbf{e}_1\mathbf{e}_1^T + \lambda_2\mathbf{e}_2\mathbf{e}_2^T + \lambda_3\mathbf{e}_3\mathbf{e}_3^T \\ &= (\lambda_1-\lambda_3)\mathbf{e}_1\mathbf{e}_1^T + (\lambda_2-\lambda_3)\mathbf{e}_2\mathbf{e}_2^T + \lambda_3\mathbf{I} = \tilde{\mathbf{T}} + \lambda_3\mathbf{I},\end{aligned} \tag{6.9}$$

where $\lambda_3\mathbf{I}$ is the isotropic part of the tensor, cf. section 5.2.3, we can see that

$$\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}} = \hat{\mathbf{v}}^T\tilde{\mathbf{T}}\hat{\mathbf{v}} + \lambda_3\hat{\mathbf{v}}^T\mathbf{I}\hat{\mathbf{v}} = \hat{\mathbf{v}}^T\tilde{\mathbf{T}}\hat{\mathbf{v}} + \lambda_3. \tag{6.10}$$

Thus it is clear that the isotropic part of the tensor can be removed without affecting the minimization problem, i.e. the minimum is obtained for the same directions. In fact it is necessary to remove it, because we will see that for computational reasons it is preferable to minimize an expression involving $\mathbf{v}$ rather than $\hat{\mathbf{v}}$.[2] Then we have the minimization of

$$\mathbf{v}^T\mathbf{T}\mathbf{v} = \mathbf{v}^T\tilde{\mathbf{T}}\mathbf{v} + \lambda_3\mathbf{v}^T\mathbf{v}, \tag{6.11}$$

which would be clearly biased against large velocities compared to (6.10). Hence we remove the isotropic part of the tensor in a preprocessing step to obtain an isotropy compensated tensor

$$\tilde{\mathbf{T}} = \mathbf{T} - \lambda_3\mathbf{I}. \tag{6.12}$$

Notice that this operation does not require a full eigenvalue decomposition of $\mathbf{T}$; it is sufficient to compute the smallest eigenvalue. An efficient algorithm for this is given in appendix G.

To simplify the notation it is understood throughout the rest of the chapter that $\mathbf{T}$ *denotes the preprocessed tensor.*

Now we can define two cost functions,

$$d_1(\mathbf{v},\mathbf{T}) = \mathbf{v}^T\mathbf{T}\mathbf{v}, \tag{6.13}$$

$$d_2(\mathbf{v},\mathbf{T}) = \frac{\mathbf{v}^T\mathbf{T}\mathbf{v}}{\|\mathbf{v}\|^2\,\mathrm{tr}\,\mathbf{T}} = \frac{\hat{\mathbf{v}}^T\mathbf{T}\hat{\mathbf{v}}}{\mathrm{tr}\,\mathbf{T}}, \tag{6.14}$$

both giving a statement about to what extent a velocity hypothesis is consistent with a given tensor. A perfect match gives the value zero, while increasing values

[2] Notice that both problems are well-defined. In the latter case we have the constraint that $\hat{\mathbf{v}}$ be of unit length while in the former case the last component of $\mathbf{v}$ has to be 1.

indicate increasing inconsistencies. The distinction between the two cost functions is that d_1 is suitable for minimization over a region, while d_2 is more useful for comparisons of the consistency of motion hypotheses at different points.

6.3.3 Parameter Estimation

Assume now that we again have a region given and that we have a motion model that assigns velocities $\mathbf{v}_i$ to each point of the region. By summing the costs at each point we obtain

$$d_{\text{tot}} = \sum_i d_1(\mathbf{v}_i, \mathbf{T}_i), \tag{6.15}$$

giving a total cost for the motion model over the entire region. With a parameterized motion model the next step is to find the parameters that minimize d_{tot}. To explain this procedure we use the affine motion model (6.5), which can be rewritten as

$$\mathbf{v} = \begin{pmatrix} v_x \\ v_y \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{S}} \underbrace{\begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \\ 1 \end{pmatrix}}_{\mathbf{p}}. \tag{6.16}$$

Hence we get

$$d_1(\mathbf{v}, \mathbf{T}) = \mathbf{v}^T \mathbf{T} \mathbf{v} = \mathbf{p}^T \mathbf{S}^T \mathbf{T} \mathbf{S} \mathbf{p} = \mathbf{p}^T \mathbf{Q} \mathbf{p}, \tag{6.17}$$

where $\mathbf{Q} = \mathbf{S}^T \mathbf{T} \mathbf{S}$ is a positive semidefinite quadratic form. Summing these over the region transforms equation (6.15) into

$$\begin{aligned} d_{\text{tot}}(\mathbf{p}) &= \sum_i d_1(\mathbf{v}_i, \mathbf{T}_i) = \sum_i \mathbf{p}^T \mathbf{S}_i^T \mathbf{T}_i \mathbf{S}_i \mathbf{p} \\ &= \mathbf{p}^T \left(\sum_i \mathbf{Q}_i \right) \mathbf{p} = \mathbf{p}^T \mathbf{Q}_{\text{tot}} \mathbf{p}, \end{aligned} \tag{6.18}$$

which should be minimized under the constraint that the last element of $\mathbf{p}$ be 1.[3] In order to do this we partition $\mathbf{p}$ and $\mathbf{Q}_{\text{tot}}$ as

$$\mathbf{p} = \begin{pmatrix} \bar{\mathbf{p}} \\ 1 \end{pmatrix}, \quad \mathbf{Q}_{\text{tot}} = \begin{pmatrix} \bar{\mathbf{Q}} & \mathbf{q} \\ \mathbf{q}^T & \alpha \end{pmatrix}, \tag{6.20}$$

[3] Now it should be clear why we prefer to minimize an expression involving $\mathbf{v}^T \mathbf{T} \mathbf{v}$ rather than $\hat{\mathbf{v}}^T \mathbf{T} \hat{\mathbf{v}}$. In the latter case equation 6.18 would be replaced by

$$d_{\text{tot}}(\mathbf{p}) = \sum_i \frac{\mathbf{p}^T \mathbf{S}_i^T \mathbf{T}_i \mathbf{S}_i \mathbf{p}}{\mathbf{p}^T \mathbf{S}_i^T \mathbf{S}_i \mathbf{p}} \tag{6.19}$$

and the minimization problem would become substantially harder to solve.

turning (6.18) into

$$d_{\text{tot}}(\mathbf{p}) = \bar{\mathbf{p}}^T \bar{\mathbf{Q}} \bar{\mathbf{p}} + \bar{\mathbf{p}}^T \mathbf{q} + \mathbf{q}^T \bar{\mathbf{p}} + \alpha. \tag{6.21}$$

If $\bar{\mathbf{Q}}$ is invertible we can complete the square to get

$$d_{\text{tot}}(\mathbf{p}) = (\bar{\mathbf{p}} + \bar{\mathbf{Q}}^{-1}\mathbf{q})^T \bar{\mathbf{Q}} (\bar{\mathbf{p}} + \bar{\mathbf{Q}}^{-1}\mathbf{q}) + \alpha - \mathbf{q}^T \bar{\mathbf{Q}}^{-1} \mathbf{q} \tag{6.22}$$

and it is clear that the minimum value

$$\alpha - \mathbf{q}^T \bar{\mathbf{Q}}^{-1} \mathbf{q} \tag{6.23}$$

is obtained for

$$\bar{\mathbf{p}} = -\bar{\mathbf{Q}}^{-1}\mathbf{q}. \tag{6.24}$$

If $\bar{\mathbf{Q}}$ should happen to be singular, the minimum value $\alpha + \mathbf{q}^T \bar{\mathbf{p}}$ is obtained for all solutions to the equation

$$\bar{\mathbf{Q}} \bar{\mathbf{p}} = -\mathbf{q} \tag{6.25}$$

and to choose between the solutions some additional constraint is needed. One reasonable possibility is to require that the mean squared velocity over the region is taken as small as possible, i.e. minimizing

$$\sum_i \bar{\mathbf{p}}^T \bar{\mathbf{S}}_i^T \bar{\mathbf{S}}_i \bar{\mathbf{p}} = \bar{\mathbf{p}}^T \left(\sum_i \bar{\mathbf{S}}_i^T \bar{\mathbf{S}}_i \right) \bar{\mathbf{p}} = \bar{\mathbf{p}}^T \mathbf{L}^2 \bar{\mathbf{p}} = \|\bar{\mathbf{p}}\|_{\mathbf{L}}, \tag{6.26}$$

where $\bar{\mathbf{S}}$ is $\mathbf{S}$ with the last column removed. The solution to this problem can be found in section 2.4.2 or in section 2.5.1 if $\mathbf{L}$ should be semidefinite.

In order to use this method of parameter estimation, the necessary and sufficient property of the motion model is that it is linear in its parameters. This property is demonstrated by equation (6.16) for the affine motion model. The corresponding matrices $\mathbf{S}$ and $\mathbf{p}$ for the constant velocity motion model (6.4) are given by

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{p} = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}, \tag{6.27}$$

and for the eight parameter motion model (6.6) by

$$\mathbf{S} = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy & 0 \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.28}$$

$$\mathbf{p} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & 1 \end{pmatrix}^T. \tag{6.29}$$

A more complex class of motion models which also are linear in their parameters, and therefore can be used in this framework, are deformable linear meshes, described by Hemmendorff [46].

There are two important advantages to estimating the velocity over a whole region rather than point by point. The first advantage is that the effects of noise

and inaccuracies in the tensor estimation typically are reduced significantly. The second advantage is that even if the aperture problem is present in some part of the region, information obtained from other parts can help to fill in the missing velocity component. There does remain a possibility that the motion field cannot be uniquely determined, but that requires the signal structures over the whole region to be oriented in such a way that the motion becomes ambiguous; a generalized aperture problem.[4] This case is characterized by $\bar{\mathbf{Q}}$ becoming singular, so that equation (6.25) has multiple solutions. The secondary requirement to minimize the mean squared velocity generalizes the idea to compute only the normal velocity component in the case of the ordinary aperture problem.

A disadvantage with velocity estimation over a whole region is that it is assumed that the true velocity field is at least reasonably consistent with the chosen motion model. A problem here is that even if we know, e.g. from the geometry of the scene, that the velocity field should be patchwise affine, we still need to obtain regions not covering patches with different motion parameters. There are many possible solutions to this problem, including graylevel segmentation and the ideal case of a priori knowledge of suitable regions. Another solution is given in the following section, where a simultaneous segmentation and velocity estimation algorithm is presented. A different alternative is to ignore the need for correct segmentation and instead simply average the $\mathbf{Q}$ matrices. This approach is described in section 6.5.

6.4 Simultaneous Segmentation and Velocity Estimation

In this section we present an efficient algorithm for simultaneous segmentation and velocity estimation, only given an orientation tensor field for one frame. The goal of the segmentation is to partition the image into a set of *disjoint* regions, so that each region is characterized by a coherent motion, with respect to the chosen motion model. In this section a region R is defined to be a nonempty, *connected* set of pixels. The segmentation algorithm is based on a competitive region growing approach. The basic algorithm is first presented in abstract form.

6.4.1 The Competitive Algorithm

To each region R is associated a cost function $C_{\mathsf{R}}(\mathbf{x})$, which is defined for all pixels in the image. Regions are extended by adding one pixel at a time. To preserve connectivity the new pixel must be adjacent to the region, and to preserve disjointedness it must not already be assigned to some other region. The new pixel is also chosen as cheap as possible. The details are as follows.

Let the border $\Delta\mathsf{R}$ of region R be the set of unassigned pixels in the image which are adjacent to some pixel in R. For each region R, the possible candidate,

[4] A nontrivial example of this generalized aperture problem is a signal consisting of concentric circles, which simultaneously expand and rotate around their center. Only the radial velocity component can be recovered.

$N(\mathsf{R})$, to be added to the region is the cheapest pixel bordering to R, i.e.

$$N(\mathsf{R}) = \arg \min_{\mathbf{x} \in \Delta \mathsf{R}} C_{\mathsf{R}}(\mathbf{x}). \tag{6.30}$$

The corresponding minimum cost for adding the candidate to the region is denoted $C_{\min}(\mathsf{R})$. In the case of an empty border, $N(\mathsf{R})$ is undefined and $C_{\min}(\mathsf{R})$ is infinite.

Assuming that a number of regions $\{\mathsf{R}_n\}$ in some way have been obtained, the rest of the image is partitioned as follows.

1. Find the region R_i for which the cost to add a new pixel is the least, i.e. $i = \arg\min_n C_{\min}(\mathsf{R}_n)$.
2. Add the cheapest pixel $N(\mathsf{R}_i)$ to R_i.
3. Repeat the first two steps until no unassigned pixels remain.

Notice that it does not matter what the actual values of the cost functions are. It is only relevant which of them is lowest. Hence the algorithm is called competitive.

6.4.2 Candidate Regions

A fundamental problem with the simultaneous segmentation and velocity estimation approach is that we typically need a segmentation in order to compute the motion model parameters, and we need motion models in order to partition the image into regions. Since we assume no a priori knowledge about the segmentation of the image, we use the concept of *candidate regions* to introduce preliminary regions into the algorithm.

To begin with we arbitrarily fill the image with a large number of overlapping rectangular candidate regions[5]. For each candidate region we then compute the optimal motion model parameters as described in section 6.3. Obviously these rectangular regions are not at all adapted to the motion field of the image and as a consequence the computed motion models are likely to be suboptimal. In order to improve the candidate regions we use a procedure called *regrowing*.

The regrowing procedure is the first application of the competitive algorithm. Regrowing is performed for one candidate region at a time, which means that there is no competition between different regions but rather between the pixels. To begin with the candidate region contains only one pixel, its *starting point*, which was also the center point of the initial rectangle. The cost function used is d_2 from equation (6.14), where $\mathbf{v}$ is the velocity given by the candidate region's current motion model. The competitive algorithm is then run until the candidate region has grown to a specified size. This size is called the *candidate region size*, m_0 and is a design parameter of the segmentation algorithm. The effect of the regrowing procedure is that the candidate region now consists of the m_0 connected pixels, starting from a fixed point, that are most consistent with the candidate

[5] e.g. squares of the size 21×21, with a distance between the center points of 4 pixels. The exact numbers are not critical.

region's motion model. When the candidate region has been regrown, new optimal parameters are computed.

Each candidate region is regrown twice, a number which seems to be sufficient to obtain reasonably coherent regions.

6.4.3 Segmentation Algorithm

Having obtained candidate regions, the rest of the segmentation algorithm is a matter of alternately converting candidate regions into real regions and letting the latter grow. In contrast to the candidate regions, the real regions are not allowed to overlap but have to be disjoint. While the candidate regions are allowed to overlap each other they must not overlap the real regions, which means that they have to be regrown from time to time, taking this restriction into account. To accomodate the inclusion of new regions, the competitive algorithm is extended to have the following steps, to be iterated as long as there are empty pixels left:

1. Regrow the candidate regions which are currently overlapping a real region. If a candidate region cannot be regrown to its full size, it is removed. The same thing happens when a candidate region's starting point becomes occupied by a real region. The cost of the most expensive included pixel is called the *maximum cost* of the candidate region.

2. Find the candidate region with the least maximum cost. This is the aspirant for inclusion among the real regions.

3. As in the competitive algorithm, find the cheapest pixel that may be added to one of the already existing real regions.

4. Compare the least maximum cost from step 2 with the cost of the cheapest pixel in step 3.

   (a) If the least maximum cost is smallest, raise the corresponding candidate region to the status of a real region.

   (b) Otherwise, add the cheapest pixel to the corresponding region.

In the first iteration there are no real regions yet, so the first thing that happens is that the best candidate region is transformed into the first real region.

To see how the segmentation algorithm works, frame 12 of the flower garden sequence, illustrated in figure 6.1, has been segmented. In figure 6.2 we can see how the regions develop and how new regions are added.

While the comparison in step 4 can be made directly between the given values, it is beneficial to introduce a design parameter λ, with which the least maximum cost is multiplied before the comparison is made. The effect of λ is that for a large value, new regions are added only if it would be very expensive to enlarge the existing ones. This may be desired e.g. if the segmentation is intended for a video coding application, where excessive fragmentation into regions can be costly. A small λ value means that existing regions are enlarged only if there are pixels available that are very consistent with the motion models, which is preferable if we

(a) frame 6 (b) frame 12 (c) frame 18

Figure 6.1: Selected frames from the flower garden sequence.

are more interested in the velocity field than in the segmentation. The difference in segmentation for varying λ values is illustrated in figure 6.3.

The regrowing of candidate regions in step 1 of the algorithm may seem prohibitively computationally expensive. In practice though, it is reasonable to assume that the maximum cost always increases when a candidate region has to be regrown.[6] Therefore it is sufficient to regrow candidate regions only when the least maximum cost is smaller than the cheapest pixel and also only a few of the top candidate regions need to be regrown.

More details on the segmentation algorithm, a few variations, and a discussion on possible improvements can be found in [21]. An algorithm with basic elements in common with the competitive algorithm can be found in [1], being applied to grayscale segmentation.[7] Initial inspiration for the development of this segmentation algorithm was given by the results in [89, 90, 91].

6.5 A Fast Velocity Estimation Algorithm

To avoid the complexities of the segmentation algorithm we may also choose to completely ignore the need for segmentation into regions with coherent motion. Instead we minimize a weighted distance measure for a motion model around each point, i.e. equation (6.18) is replaced by

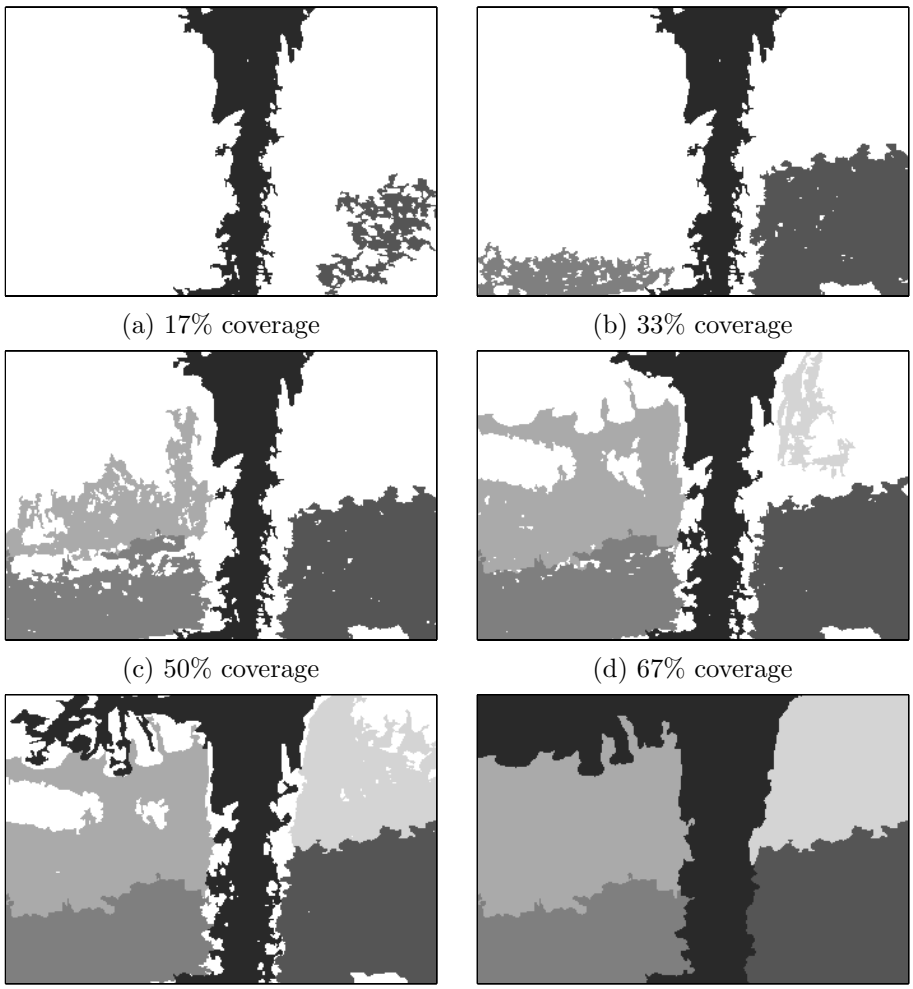
$$d_{\text{tot}}(\mathbf{p}) = \sum_i w_i d_1(\mathbf{v}_i, \mathbf{T}_i) = \mathbf{p}^T \left(\sum_i w_i \mathbf{Q}_i \right) \mathbf{p} = \mathbf{p}^T \mathbf{Q}_{\text{tot}} \mathbf{p}, \tag{6.31}$$

where the sum is taken over a neighborhood of the current point and the weights w_i are given by, e.g., a Gaussian. In effect this means that we convolve the quadratic forms $\mathbf{Q}_i$ over the image with the weight function, and this operation can be efficiently computed by separable convolution as soon as the weight function is separable. Another way to look at this operation is as an application of normalized averaging, see section 3.9.1, with the weight function as applicability.[8] By taking

[6] This would have been strictly correct if the motion model parameters were not recomputed each time the candidate region is regrown.

[7] The competitive algorithm presented here was developed independently, although being predated by the mentioned paper.

[8] Notice that this gives a somewhat different scaling of the results, especially at the borders.

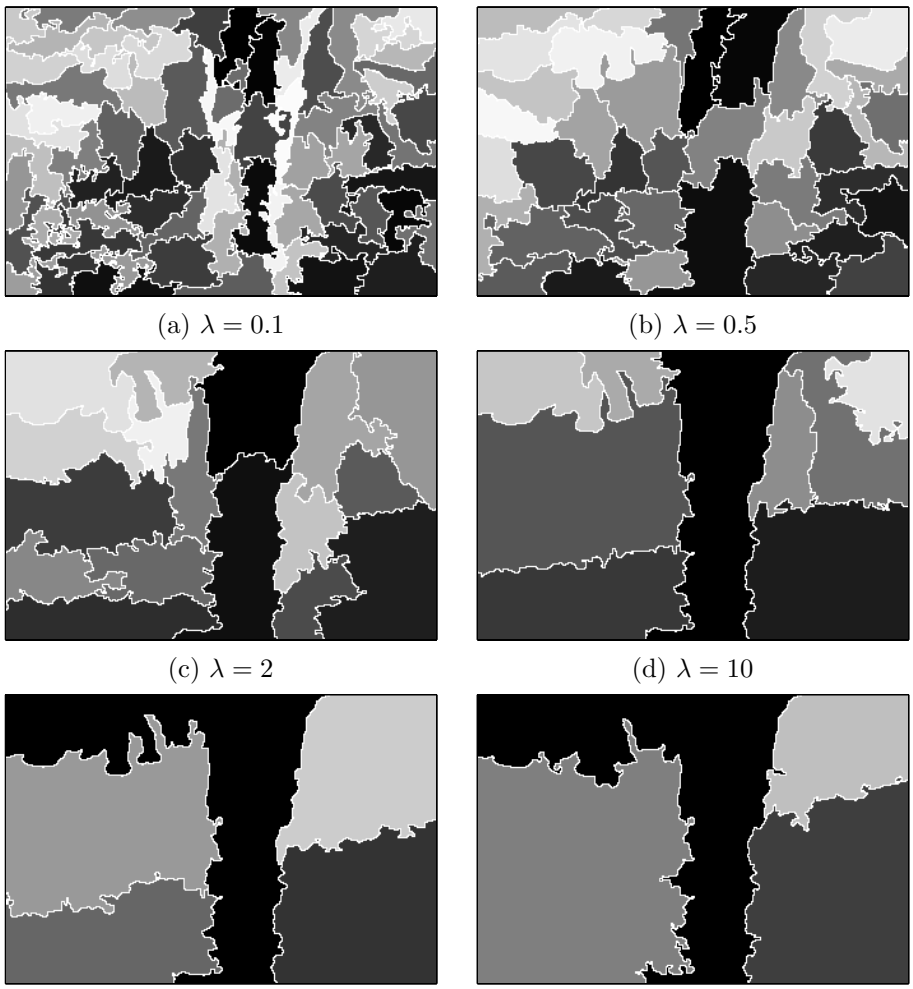


(a) 17% coverage (b) 33% coverage

(c) 50% coverage (d) 67% coverage

(e) 83% coverage (f) 100% coverage

Figure 6.2: Development of the regions in the segmentation algorithm.



(a) $\lambda = 0.1$ (b) $\lambda = 0.5$

(c) $\lambda = 2$ (d) $\lambda = 10$

(e) $\lambda = 50$ (f) $\lambda = 500$

Figure 6.3: Segmentation results for different λ values.

this view we have the opportunity to set the certainty field to zero close to the borders, which is appropriate if we only use the very fast separable correlation method from section 4.3 to compute the tensors, as that method gives incorrect results at the borders.

The optimal parameters and the corresponding velocity estimates at each point are computed exactly as in section 6.3.3 and it also turns out that the minimum value (6.23) can be used as a confidence measure,[9] since it indicates how well the local neighborhood is consistent with the motion model in use.

In the simplest case of a constant velocity motion model, we have $\mathbf{S} = \mathbf{I}$ and hence the averaging of the quadratic forms $\mathbf{Q}_i$ reduces to an averaging of the tensor field. This is in fact a well known idea to improve the robustness of the tensors [94], but there are a few important differences. The first one is that here we do not average the original tensor field, but rather the isotropy compensated field. The second difference is that we compute the velocity by equation (6.24), solving an equation system, rather than by (6.2) or (6.3), which involves the computation of at least one eigenvector.

To summarize the whole algorithm we have the following five steps:

1. Compute the orientation tensor field for the frame, preferably using the separable correlation method for maximum computational efficiency.
2. Remove the isotropic part of the tensors.
3. Compute quadratic forms, $\mathbf{Q}_i = \mathbf{S}_i^T \mathbf{T}_i \mathbf{S}_i$, according to the chosen motion model.
4. Apply normalized averaging to the quadratic forms.
5. Solve for the optimal parameters $\mathbf{p}_i$ and compute the corresponding velocity estimates $\mathbf{v}_i = \mathbf{S}_i \mathbf{p}_i$.

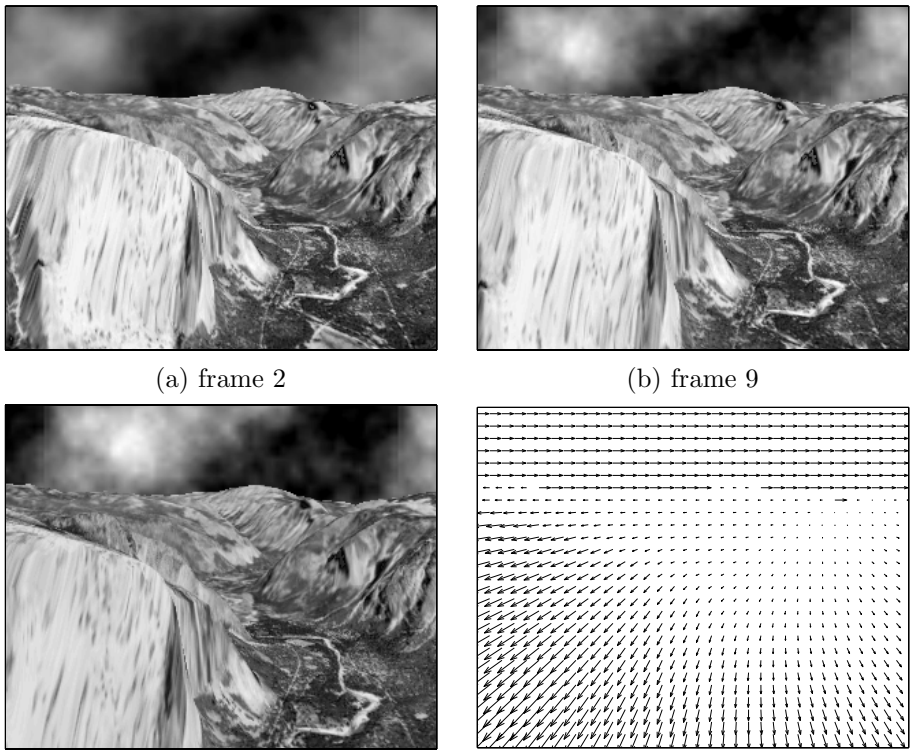
6.6 Evaluation

The velocity estimation algorithms have been evaluated on two commonly used test sequences with known velocity fields, Lynn Quam's Yosemite sequence [45], figure 6.4, and David Fleet's diverging tree sequence [31], figure 6.5. Both sequences are synthetic but differs in that the Yosemite sequence is generated with the help of a digital terrain map and therefore has a motion field with depth variation and discontinuities at occlusion boundaries. The diverging tree sequence on the other hand is only a textured planar surface towards which the camera translates. Hence the motion field is very regular but the lack of image details leads to difficulties in the velocity estimation.

Additionally the algorithms have been tested on three real image sequences without known velocity fields. A few more evaluation results can be found in the report from the ICPR 2000 Algorithm Performance Contest [2] but are not reproduced here.[10]

[9] Actually it is a reversed confidence measure since small values indicate high confidence.

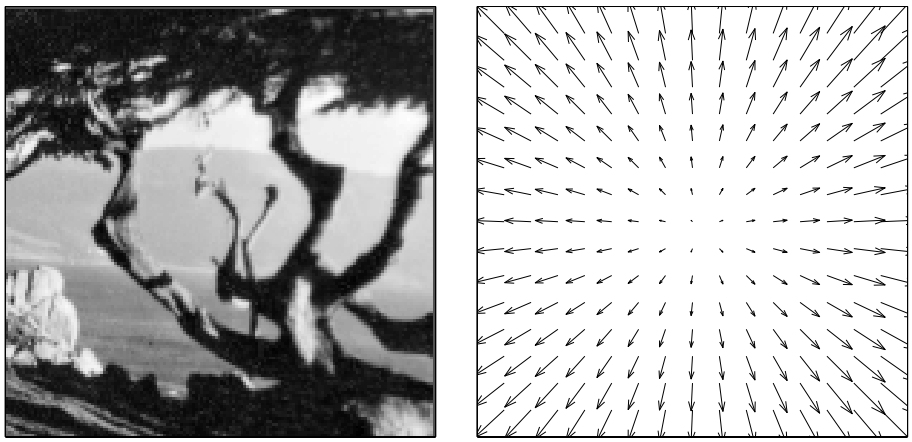
[10] For readers of the report it may be useful to know that the false alarm rate counts the fraction



(a) frame 2 (b) frame 9

(c) frame 16 (d) velocity field

Figure 6.4: Selected frames from the Yosemite sequence and the true velocity field corresponding to frame 9 (subsampled).



(a) frame 20 (b) velocity field

Figure 6.5: One frame from the diverging tree sequence and the corresponding true velocity field (subsampled).

6.6.1 Implementation and Performance

All algorithms have been implemented in Matlab, with Normalized Convolution, ordinary convolution/correlation, the segmentation algorithm, and solution of multiple equation systems in the form of C mex files and the rest as highly vectorized matlab code.

Typical running times for the different algorithms on a 360 MHz SUN Ultra 60 are given below and relate to the computation of the velocity for one frame of the 252×316 Yosemite sequence.

Velocity estimation with the segmentation algorithm takes about 30 seconds, distributed with 1.6 seconds for tensor estimation with the separable convolution method, 1.1 seconds for estimation of the tensors along the border with the separable normalized convolution method, 0.5 seconds for isotropy compensation, and 27 seconds for the segmentation algorithm, most of which is spent on the construction of candidate regions. Here the affine motion model is used, effective size of the kernels in the tensor estimation is $9 \times 9 \times 9$, and candidate region size m_0 is 500.

The fast algorithm with the affine motion model, $11 \times 11 \times 11$ tensors, and a 41×41 averaging kernel takes about 16 seconds. Of these, 1.8 seconds are spent on tensor estimation with the separable convolution method, 0.5 seconds on isotropy compensation, 0.3 seconds on computation of the quadratic forms, 8.6 seconds on normalized averaging, and 4.8 seconds on solving for the velocity.

Finally we have the fast algorithm with the constant velocity motion model, $9 \times 9 \times 9$ tensors and 15×15 normalized averaging. Here the running time is

of non-zero velocity estimates in the uniformly gray background of the test sequences. We leave it to the readers to decide whether this is a useful measure or relevant to their applications.

about 3.5 seconds, with 1.6 seconds for tensor estimation, 0.5 seconds for isotropy compensation, 1.2 seconds for normalized averaging, and 0.2 seconds for solving for the velocity.

6.6.2 Results for the Yosemite Sequence

The accuracy of the velocity estimates has been measured using the average spatiotemporal angular error, $\arccos(\hat{\mathbf{v}}_{\text{est}}^T \hat{\mathbf{v}}_{\text{true}})$ [8]. In the Yosemite sequence the sky region is excluded from the error analysis.[11]

To see how the various design parameters affect the results, we present a fairly detailed analysis. Common parameters for all algorithms are the values of σ and γ in the tensor estimation, cf. sections 5.7.2 and 5.7.3.[12] Additional parameters for the segmentation algorithm are the factor λ and the candidate region size m_0. The fast algorithm only adds the standard deviation σ_{avg} for the averaging kernel, which is chosen to be Gaussian. The kernel sizes used by the various algorithms are the same as in the discussion on running times.

For the segmentation algorithm, we begin by varying m_0 while having $\sigma = 1.4$, $\gamma = \frac{1}{8}$, and $\lambda = 0.06$. The results are shown in figure 6.6(a) and we can see that the errors vary between 1.25° and 1.45° in a rather unpredictable way. The main reason for this peculiar phenomenon is that the final partitioning into regions as well as the motion model parameters, which are computed only from the initial pixels in the region, can change significantly with a small change in m_0. In figure 6.6(b)–(d) we plot the minimum, mean, and maximum values for the average angular errors over the interval $400 \leq m_0 \leq 600$, while in turn varying σ, γ, and λ around the values given above.

While the sensitivity to the value of m_0 is disturbing, it turns out that this problem can be eliminated nicely at the cost of some extra computation. The solution is to estimate the velocity for a number of different values of m_0 and then simply average the estimates.[13] This has the double effect of both stabilizing the estimates and improving them. Using 11 evenly spaced values $400, 420, \ldots, 600$ of m_0 we get an average angular error of 1.14° and a standard deviation of 2.14°. Picking 11 m_0 values randomly in the same interval, we consistently get average angular errors between 1.13° and 1.18°.

In figure 6.7(a)–(c) we see the results for the fast algorithm with the affine motion model, in turn varying σ, γ and σ_{avg} around the point $\sigma = 1.6$, $\gamma = \frac{1}{256}$ and $\gamma_{\text{avg}} = 6.5$. Of interest here is that a large part of the errors are due to discontinuities in the velocity field, especially along the horizon. It turns out that the confidence measure is rather successful in identifying the uncertain estimates. Sorting the estimates with respect to the confidence, we can compute average angular errors at different levels of coverage, shown in figure 6.7(d). At 100%

[11] Notice that the sky region is only excluded from the error analysis, however. It is not blanked out or otherwise removed prior to estimating the velocity field.

[12] We only consider Gaussian applicabilities. To some extent the kernel size may also be regarded as a design parameter, but its only effect is a trade-off between the computation time and the usable range for σ.

[13] Notice that we only need to rerun the segmentation algorithm. The tensor field can be reused. The running time is thus increased to about five minutes when 11 m_0 values are used.

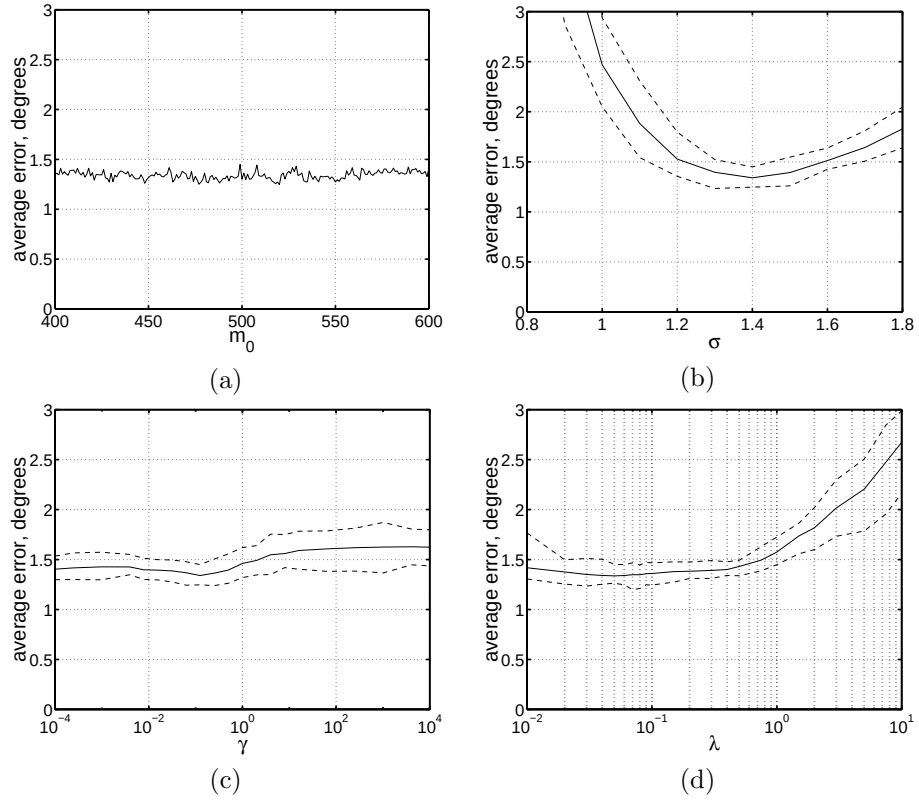


Figure 6.6: Average angular errors for the segmentation algorithm on the Yosemite sequence, while varying the design parameters.

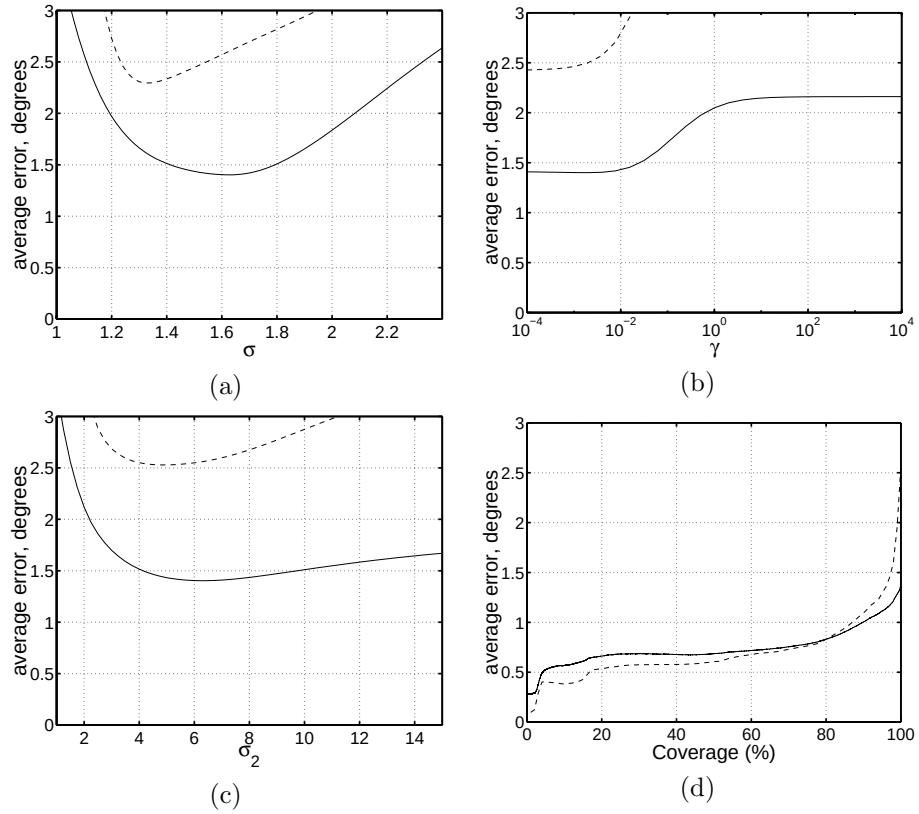


Figure 6.7: (a)–(c): Average angular errors for the fast algorithm with the affine motion model on the Yosemite sequence, while varying the design parameters. (d): Average angular errors at different levels of coverage. The dashed lines are the corresponding standard deviations.

coverage we have an average angular error of $1.40° \pm 2.57°$, at 90% the error is $1.00° \pm 1.09°$, and at 70% it is $0.75° \pm 0.73°$.[14]

Using the constant velocity motion model instead of affine motion we obtain average angular errors at different levels of coverage according to figure 6.8 for $\sigma = 1.4$, $\gamma = \frac{1}{32}$, and $\sigma_{\text{avg}} = 3.5$. The errors are increased to $1.94° \pm 2.31°$ at 100% coverage, $1.61° \pm 1.57°$ at 90%, and $1.43° \pm 1.24°$ at 70%. These results can be compared to results for a similar simple method, reported by Karlholm in [61]. He uses orientation tensors estimated from quadrature filter responses at multiple scales. From these the isotropic part is removed and they are normalized with respect to the largest eigenvalue. Finally the squared tensors are averaged using a 21×21 Gaussian kernel with standard deviation 6 and the velocity is

[14] $a \pm b$ is used here as a shorthand for average error and standard deviation, but has no meaningful interpretation in terms of an interval.

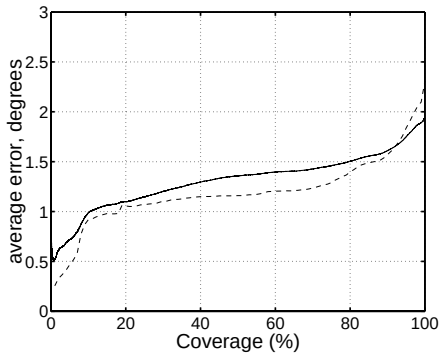


Figure 6.8: Average angular errors for the fast algorithm with the constant velocity motion model on the Yosemite sequence at different levels of coverage. The dashed line gives the corresponding standard deviations.

		segmentation	fast, affine	fast, constant
Average error		1.14°	1.40°	1.94°
Standard deviation		2.14°	2.57°	2.31°
Proportion of estimates with errors below:	$< 0.5°$	32.0%	35.8%	14.1%
	$< 1°$	64.4%	65.0%	39.7%
	$< 2°$	87.8%	82.1%	70.5%
	$< 3°$	94.0%	89.7%	83.4%
	$< 5°$	98.0%	95.4%	92.8%
	$< 10°$	99.7%	98.8%	98.6%

Table 6.1: Distribution of errors for the Yosemite sequence.

estimated from the smallest eigenvector as in equation (6.3). This gives average angular errors of $2.44° \pm 2.06°$ at 90% coverage and $2.23° \pm 1.94°$ at 70%, using the quotient $\frac{\lambda_2}{\lambda_1}$ as confidence measure.

It would also be conceivable to use the eight parameter motion model with the fast algorithm but it turns out to give no better results than the affine motion model. In fact the results are slightly worse, probably due to model overfitting.[15]

Some statistics on the distribution of errors for the three evaluated methods are given in table 6.1. Comparison with previously published results, table 6.2, shows that the algorithms presented here are substantially[16] more accurate than existing methods.

[15]The constant velocity motion model and the eight parameter motion model can of course be used with the segmentation algorithm too, but do not lead to any improvements for this sequence.

[16]The margins are, however, considerably smaller than when this comparison was made in [23].

Technique	Average error	Standard deviation	Density
Lucas & Kanade [71]	2.80°	3.82°	35%
Uras *et al.* [87]	3.37°	3.37°	14.7%
Fleet & Jepson [31]	2.97°	5.76°	34.1%
Xu [100]	4.90°	7.34°	99.8%
Black & Anandan [12]	4.46°	4.21°	100%
Szeliski & Coughlan [81]	2.45°	3.05°	100%
Black & Jepson [13]	2.29°	2.25°	100%
Ju *et al.* [59]	2.16°	2.0°	100%
Karlholm [61]	2.06°	1.72°	100%
Lai & Vemuri [69]	1.99°	1.41°	100%
Bab-Hadiashar & Suter [7]	1.97°	1.96°	100%
Mémin & Pérez [73]	1.58°	1.21°	100%
segmentation	1.14°	2.14°	100%
fast, affine	1.40°	2.57°	100%
fast, affine	0.75°	0.73°	70%
fast, constant	1.94°	2.31°	100%
fast, constant	1.43°	1.24°	70%

Table 6.2: Comparison of error results for the Yosemite sequence. All errors are computed without the sky region. The compilation of the old results is based on a similar table by Karlholm [61].

Technique	Average error	Standard deviation	Density
Horn & Schunck (modified)	2.55°	3.67°	100%
Lucas & Kanade	1.65°	1.48°	24.3%
Uras *et al.*	3.83°	2.19°	60.2%
Nagel	2.94°	3.23°	100%
Fleet & Jepson	0.73°	0.46°	28.2%
Liu *et al.* [70]	1.86°	1.35°	100%
Xu [100]	4.11°	6.56°	93.7%
Gökstorp [34]	0.94°	0.60°	50.0%
segmentation	0.54°	0.28°	100%
fast, affine	0.56°	0.23°	100%
fast, constant	1.79°	1.34°	100%

Table 6.3: Comparison of error results for the diverging tree sequence.

6.6.3 Results for the Diverging Tree Sequence

The diverging tree sequence is characterized by having a continuous velocity field, in contrast to the discontinuities in the Yosemite sequence. On the other hand there is less texture and large regions which are completely featureless. One result of these changed circumstances is that the confidence measure for the fast method turns ineffective, since the larger estimation errors mainly are caused by a lack of image details rather than incoherency in the local velocity field.[17] Hence no results are given for partial levels of coverage.

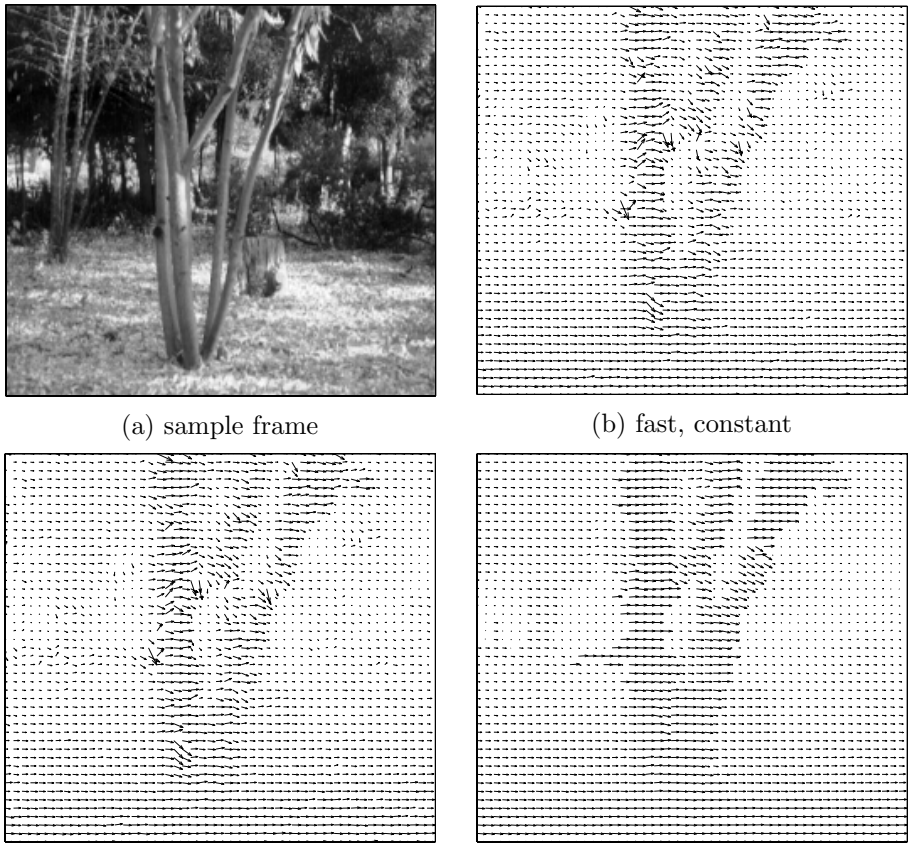
The segmentation algorithm, with $\sigma = 1.25$, $\gamma = \frac{1}{8}$, $\lambda = 0.25$, and the velocity averaged over $m_0 = 500, 520, \ldots, 700$ gives an average angular error of 0.54° and a standard deviation of 0.28°. The fast algorithm with the affine motion model and $\sigma = 1.6$, $\gamma = \frac{1}{4}$, and $\sigma_{\text{avg}} = 9.5$ (51×51 Gaussian kernel) gives an average angular error of $0.56° \pm 0.23°$. The fast algorithm with the constant velocity motion model and $\sigma = 1.1$, $\gamma = \frac{1}{32}$, and $\sigma_{\text{avg}} = 1.5$ results in an average angular error of $1.79° \pm 1.34°$.

That the segmentation algorithm gives only marginally better results than the fast algorithm with the affine motion model is not surprising given the lack of discontinuities in the velocity field. A comparison with other methods is given in table 6.3. Entries without an explicit citation are from [8].

6.6.4 Results for Real Image Sequences

The velocity estimation algorithms have also been tested on three real image sequences also used in [8]. No ground truth data are available for these and therefore we only present arrow plots of the estimated velocity fields in figures 6.9–6.11.

[17] Necessary information to detect this kind of uncertainty should be available in the tensor field.

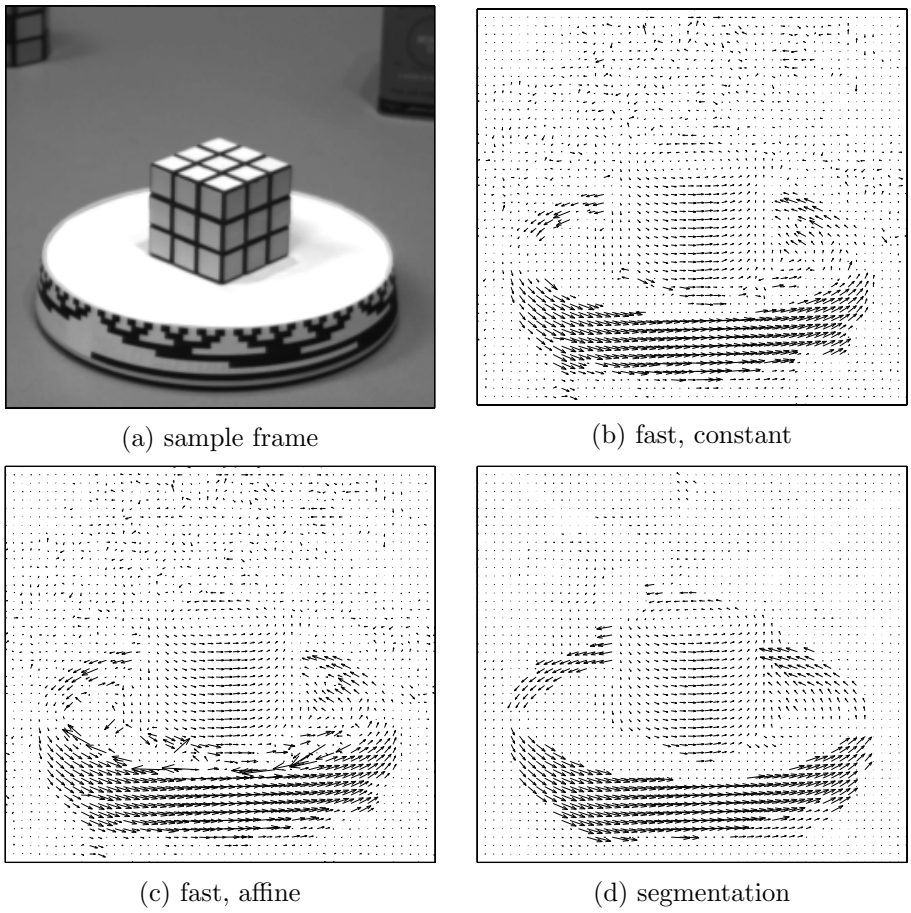


(a) sample frame (b) fast, constant

(c) fast, affine (d) segmentation

Figure 6.9: SRI Trees sequence. Sample frame and estimated velocity fields (subsampled).

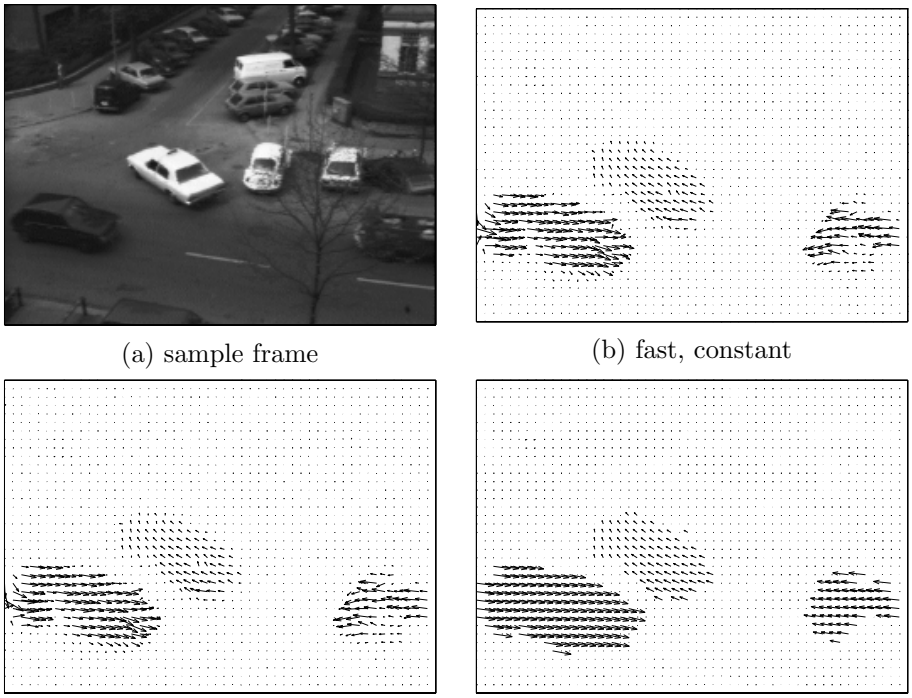
The first velocity field in each figure is estimated by the fast algorithm with the constant motion model, the second field by the fast algorithm with the affine motion model, and the third field by the segmentation algorithm with the affine motion model. In all cases $9 \times 9 \times 9$ tensors with $\sigma = 1.2$ have been used. The averaging for the fast algorithm has been done over 11×11 areas with $\sigma_{\text{avg}} = 1.5$ for the constant motion model and over 13×13 areas with $\sigma_{\text{avg}} = 1.8$ for the constant motion model. The segmentation algorithm uses the parameters $m_0 = 400$ and $\lambda = 0.6$.



(a) sample frame (b) fast, constant

(c) fast, affine (d) segmentation

Figure 6.10: Rubik Cube sequence. Sample frame and estimated velocity fields (subsampled).



(a) sample frame (b) fast, constant

(c) fast, affine (d) segmentation

Figure 6.11: Hamburg Taxi sequence. Sample frame and estimated velocity fields (subsampled).

Chapter 7

Displacement Estimation

7.1 Introduction

A limitation of the motion estimation algorithms in chapter 6, and methods based on spatiotemporal filtering in general, is that they require the motion field to be consistent over several frames for good results. Unfortunately this is not always the case.

This limitation has turned out to be a real problem in the WITAS project (see section 1.1), where image sequences are obtained by a helicopter-mounted camera. Six consecutive frames from a test flight at Revinge are shown in figure 7.1. The camera system is affected by vibrations from the helicopter, causing small but still significant displacements of the images at each frame. Moreover these displacements change very quickly and are therefore difficult to predict or compensate for. Figure 7.2 shows the estimated displacement fields between successive frames. These include motion induced both by the regular movement of the helicopter and by vibrations. Since the former should be mostly constant within the short sequence of frames involved here, it is clear that the vibrations are indeed significant.

One possible solution to the problem with vibrations is to only use two frames and estimate the displacement between these. Then the background is assumed to move according to a parametric model, which allows us to estimate and compensate for the ego-motion, including vibrations. This chapter introduces a novel two-frame displacement estimation algorithm, based on polynomial expansion, and develops the necessary machinery to detect moving objects in image sequences of the type shown in figure 7.1. The final detection results for that sequence can be found in section 7.10.

7.2 Displacement of a Quadratic Polynomial

Since the result of polynomial expansion is that each neighborhood is approximated by a polynomial, it is interesting to analyze what happens if a polynomial

(a)

(b)

(c)

(d)

(e)

(f)

Figure 7.1: Six consecutive frames from a test flight at Revinge. Subsampled a factor two from the original video sequence.

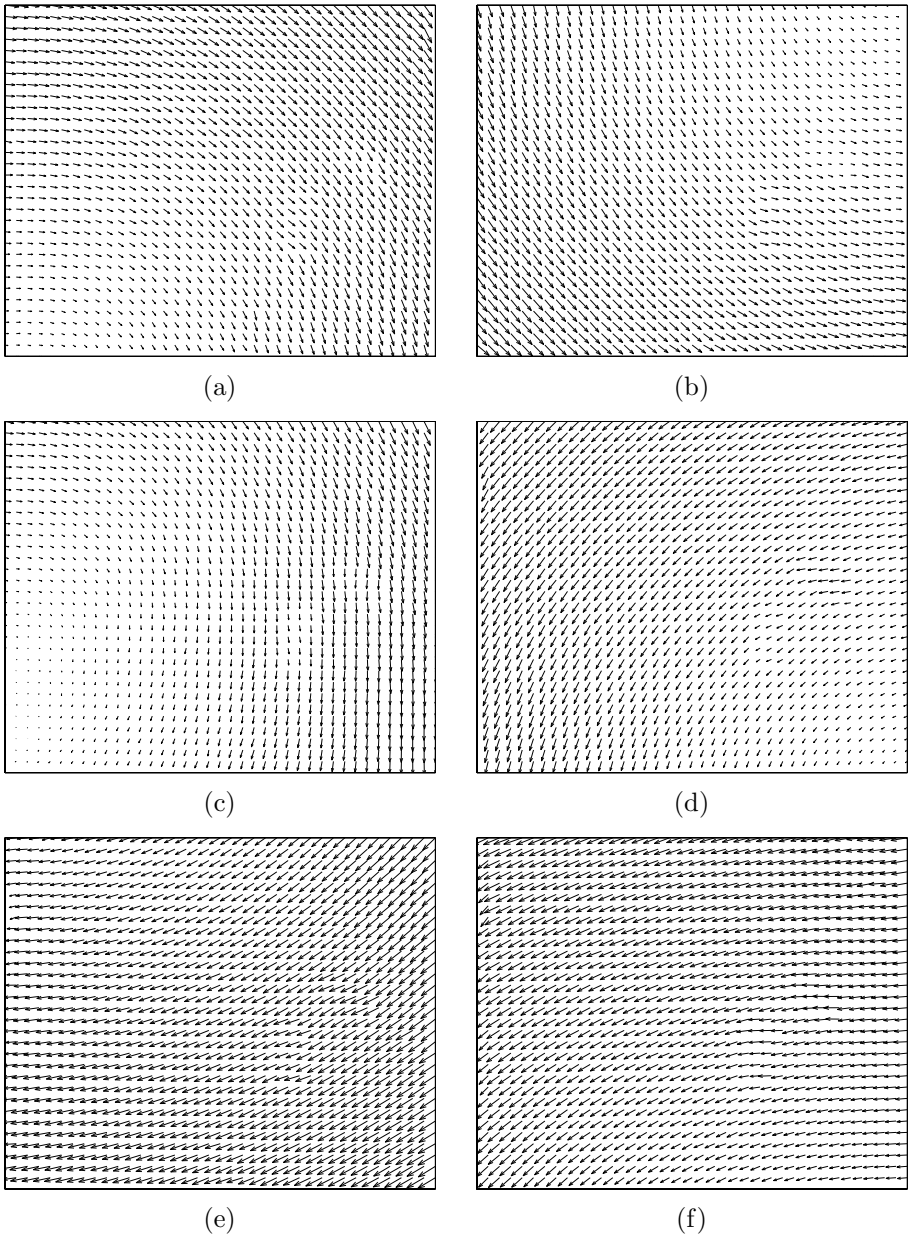


Figure 7.2: Estimated displacement fields between pairs of frames in figure 7.1. The maximum displacements are about 4.5 pixels.

undergoes an ideal translation.

Consider the exact quadratic polynomial

$$f_1(\mathbf{x}) = \mathbf{x}^T\mathbf{A}_1\mathbf{x} + \mathbf{b}_1^T\mathbf{x} + c_1 \tag{7.1}$$

and construct a new signal f_2 by a global displacement by $\mathbf{d}$,

$$\begin{aligned} f_2(\mathbf{x}) &= f_1(\mathbf{x}-\mathbf{d}) \\ &= (\mathbf{x}-\mathbf{d})^T\mathbf{A}_1(\mathbf{x}-\mathbf{d}) + \mathbf{b}_1^T(\mathbf{x}-\mathbf{d}) + c_1 \\ &= \mathbf{x}^T\mathbf{A}_1\mathbf{x} - 2\mathbf{d}^T\mathbf{A}_1\mathbf{x} + \mathbf{d}^T\mathbf{A}_1\mathbf{d} + \mathbf{b}_1^T\mathbf{x} - \mathbf{b}_1^T\mathbf{d} + c_1 \\ &= \mathbf{x}^T\mathbf{A}_1\mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d})^T\mathbf{x} + \mathbf{d}^T\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^T\mathbf{d} + c_1 \\ &= \mathbf{x}^T\mathbf{A}_2\mathbf{x} + \mathbf{b}_2^T\mathbf{x} + c_2. \end{aligned} \tag{7.2}$$

Equating the coefficients in the quadratic polynomials yields

$$\mathbf{A}_2 = \mathbf{A}_1, \tag{7.3}$$

$$\mathbf{b}_2 = \mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d}, \tag{7.4}$$

$$c_2 = \mathbf{d}^T\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^T\mathbf{d} + c_1. \tag{7.5}$$

The key observation is that by equation (7.4) we can solve for the translation $\mathbf{d}$, at least if $\mathbf{A}_1$ is non-singular,

$$2\mathbf{A}_1\mathbf{d} = -(\mathbf{b}_2 - \mathbf{b}_1), \tag{7.6}$$

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}_1^{-1}(\mathbf{b}_2 - \mathbf{b}_1). \tag{7.7}$$

We note that this observation holds for any signal dimensionality.

7.2.1 Intuitive Explanation

The stationary points of

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \tag{7.8}$$

can be found by differentiating f and setting the result to 0,

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{0}, \tag{7.9}$$

$$\mathbf{x} = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}. \tag{7.10}$$

If we assume that $\mathbf{A}$ is non-singular and rewrite (7.7) as

$$\mathbf{d} = (-\frac{1}{2}\mathbf{A}_2^{-1}\mathbf{b}_2) - (-\frac{1}{2}\mathbf{A}_1^{-1}\mathbf{b}_1), \tag{7.11}$$

we obtain the displacement as the observed movement of the stationary point.

7.2.2 Practical Considerations

Obviously the assumptions about an entire signal being a single polynomial and a global translation relating the two signals are quite unrealistic. Still the basic relation (7.6) can be used for real signals, although errors are introduced when the assumptions are relaxed. The question is whether these errors can be kept small enough to give useful algorithms.

To begin with we replace the global polynomial in equation (7.1) with local polynomial approximations. Thus we start by doing a polynomial expansion of both images, giving us expansion coefficients $\mathbf{A}_1(\mathbf{x})$, $\mathbf{b}_1(\mathbf{x})$, and $c_1(\mathbf{x})$ for the first image and $\mathbf{A}_2(\mathbf{x})$, $\mathbf{b}_2(\mathbf{x})$, and $c_2(\mathbf{x})$ for the second image. Ideally this should give $\mathbf{A}_1 = \mathbf{A}_2$ according to equation (7.3) but in practice we have to settle for the approximation

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\mathbf{x})}{2}. \tag{7.12}$$

We also introduce

$$\Delta\mathbf{b}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2(\mathbf{x}) - \mathbf{b}_1(\mathbf{x})) \tag{7.13}$$

to obtain the primary constraint

$$\mathbf{A}(\mathbf{x})\mathbf{d}(\mathbf{x}) = \Delta\mathbf{b}(\mathbf{x}), \tag{7.14}$$

where $\mathbf{d}(\mathbf{x})$ indicates that we have also replaced the global displacement in equation (7.2) with a spatially varying displacement field.

The first application of these observations is an extremely simple but functional and fast disparity estimation algorithm.

7.3 A Simple Disparity Estimation Algorithm

The assumption is that we have a stereo pair of two images, called left and right, which are related by a spatially varying displacement field, where all displacements, the disparities, are along the x-axis. For the standard stereo geometry (see appendix H) with two parallel pinhole cameras, the disparities are inversely proportional to the depths in the scene. To illustrate the algorithm we use a stereo pair from the well-known SRI Trees sequence, shown in figure 7.3.

Using indices r and l for the right and left images instead of 1 and 2, equations (7.12) and (7.13) turn into

$$\mathbf{A}(x, y) = \frac{\mathbf{A}_r(x, y) + \mathbf{A}_l(x, y)}{2}, \tag{7.15}$$

$$\Delta\mathbf{b}(x, y) = -\frac{1}{2}(\mathbf{b}_l(x, y) - \mathbf{b}_r(x, y)). \tag{7.16}$$

In principle we should now be able to obtain a disparity estimate at (x, y) by solving equation (7.14) pointwise, i.e.

$$\mathbf{d}(x, y) = \mathbf{A}(x, y)^{-1}\Delta\mathbf{b}(x, y). \tag{7.17}$$

Figure 7.3: Stereo pair.

Notice that this gives a 2D displacement $\mathbf{d}$, which may have a non-zero y component. Since the disparities are assumed to be limited to the x direction, we can use the relative size of the y component as a confidence measure.

Unfortunately, and not very surprising, these estimates turn out to be too noisy and uncertain to be really useful. There is also the problem that $\mathbf{A}(x, y)$ may be singular or close to singular. The result of applying this operation to the stereo pair in figure 7.3 is shown in figure 7.4.

If we make the assumption that the disparity field is only slowly varying, we can improve the estimates through an averaging operation. The drawback is that step discontinutities in the disparity field will be smoothed out.

Figure 7.4: Disparity estimates from equation (7.17). Values outside the interval $[0, 6]$ have been truncated. Disparity 0 is shown as black and 6 as white.

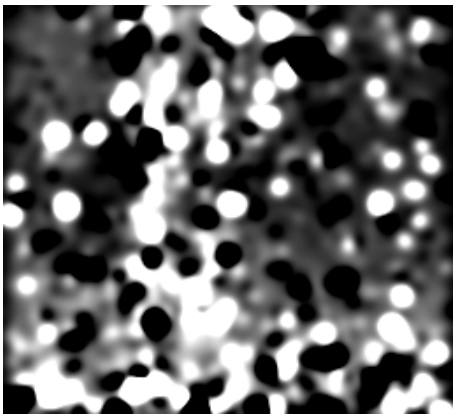


Figure 7.5: Local averages of the disparities shown in figure 7.4.

The naive way to implement the averaging would be to simply compute a local average of the pointwise disparity estimates obtained through equation (7.17), e.g. by filtering them with some lowpass filter. As we can see in figure 7.5, this gives a very unsatisfactory result. The main reason is that we have occasional disparity estimates which are way off, possibly several thousand pixels large or more.

A better solution would be to apply the averaging implicitly to the constraints (7.14). Such an approach is used in section 7.4 for general displacement estimation. Here we opt for simplicity and instead explore what can be done with normalized averaging (see section 3.9.1).

We have three sources of certainty information. As has already been remarked, equation (7.17) yields a 2D displacement field, although we know a priori that the y component should be zero. Thus we introduce

$$c_1(x,y) = \frac{d_x(x,y)^2}{d_x(x,y)^2 + d_y(x,y)^2} \tag{7.18}$$

as a measure of the relative sizes of the two components.

The second certainty source is based on the observation in appendix H that knowledge of the minimum and maximum depth in the scene allows establishing a priori bounds on the disparity. We set certainty to zero for all outliers,

$$c_2(x,y) = \begin{cases} 1, & d_{\min} \le d_x(x,y) \le d_{\max}, \\ 0, & \text{otherwise.} \end{cases} \tag{7.19}$$

Although we do not know the actual camera parameters or depth bounds for the stereo pair in figure 7.3, we can by observation estimate $d_{\min} = 0$ and $d_{\max} = 6$.

The last source of certainty is related to the computation of the polynomial expansion. If this is done with the fastest algorithm, separable correlation (SC), we get unreliable expansion coefficients close to the edges. Assuming that the

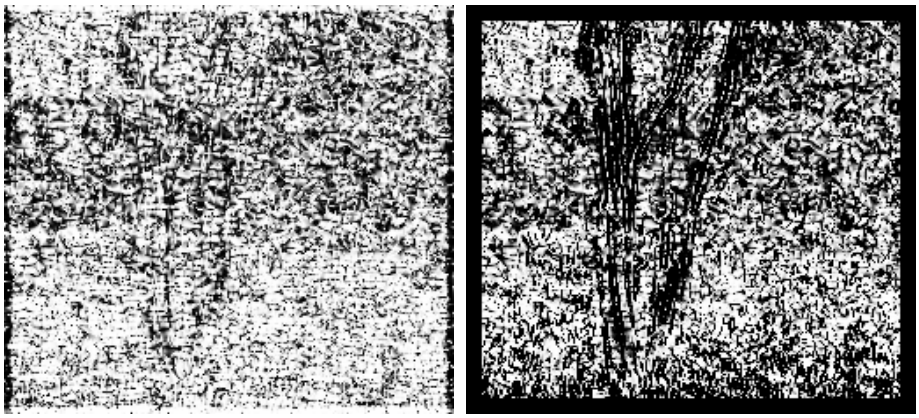


Figure 7.6: Certainty fields c_1 and c. Black corresponds to certainty 0 and white to certainty 1.

applicability used for polynomial expansion was of size $N \times N$, we set

$$c_3(x,y) = \begin{cases} 0, & (x,y) \text{ within } \frac{N-1}{2} \text{ pixels from the edge,} \\ 1, & \text{otherwise.} \end{cases} \tag{7.20}$$

If the polynomial expansion is done by either of the normalized convolution (NC) or separable normalized convolution (SNC) methods, there is still reason to lower the certainty at the very edges but not as drastically as here.

The total certainty is computed as the product of these three,

$$c(x,y) = c_1(x,y)c_2(x,y)c_3(x,y). \tag{7.21}$$

The first certainty component c_1 and the total certainty c are shown in figure 7.6.

The signal used in the normalized averaging is of course the disparity estimates, i.e. the x component of the displacements computed by equation (7.17), and we use a Gaussian applicability. The result can be seen in figure 7.7. It is certainly not perfect, but adequate considering that the primary design goal of the algorithm was simplicity.

The final algorithm is summarized below:

1. Compute polynomial expansions $\mathbf{A}_l$, $\mathbf{b}_l$, c_l and $\mathbf{A}_r$, $\mathbf{b}_r$, c_r for the left and right images respectively.

2. Compute $\mathbf{A}$ and $\boldsymbol{\Delta}\mathbf{b}$ according to equations (7.15) and (7.16).

3. Compute displacement vectors $\mathbf{d}$ by solving 2×2 equation systems according to equation (7.17).

4. Compute certainty values according to equations (7.18)–(7.21).

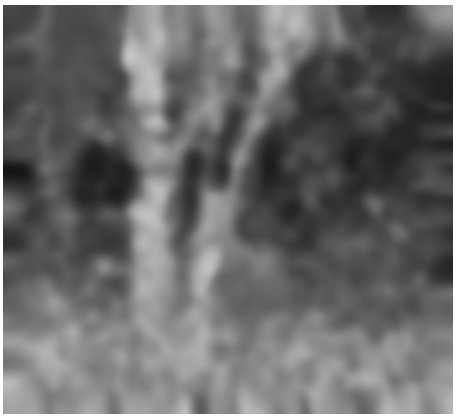


Figure 7.7: Disparity estimates after normalized averaging.

5. Apply normalized averaging to the x component of the displacement computed in step 3, using a Gaussian applicability and c as certainty. This gives the final disparity estimates.

7.4 Displacement Estimation

In this section we go back to the problem of estimating a general displacement field, which is not constrained to the x-axis. As we saw in the previous section, pointwise solution of (7.14) does not give good results. To improve this we make, as before, the assumption that the displacement field is only slowly varying. This time we try to find $\mathbf{d}(\mathbf{x})$ satisfying (7.14) as well as possible over a neighborhood I of $\mathbf{x}$, or more formally minimizing

$$\sum_{\boldsymbol{\Delta}\mathbf{x}\in I} w(\boldsymbol{\Delta}\mathbf{x})\|\mathbf{A}(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x})\mathbf{d}(\mathbf{x}) - \boldsymbol{\Delta}\mathbf{b}(\mathbf{x}+\boldsymbol{\Delta}\mathbf{x})\|^2, \tag{7.22}$$

where we let $w(\boldsymbol{\Delta}\mathbf{x})$ be a weight function (applicability). The minimum is obtained for

$$\mathbf{d}(\mathbf{x}) = \Big(\sum w\mathbf{A}^T\mathbf{A}\Big)^{-1}\sum w\mathbf{A}^T\boldsymbol{\Delta}\mathbf{b}, \tag{7.23}$$

where we have dropped some indexing to make the expression more readable. The minimum value is given by

$$e(\mathbf{x}) = \Big(\sum w\boldsymbol{\Delta}\mathbf{b}^T\boldsymbol{\Delta}\mathbf{b}\Big) - \mathbf{d}(\mathbf{x})^T\sum w\mathbf{A}^T\boldsymbol{\Delta}\mathbf{b}. \tag{7.24}$$

In practical terms this means that we compute $\mathbf{A}^T\mathbf{A}$, $\mathbf{A}^T\boldsymbol{\Delta}\mathbf{b}$, and $\boldsymbol{\Delta}\mathbf{b}^T\boldsymbol{\Delta}\mathbf{b}$ pointwise and average these with w before we solve for the displacement. The minimum value $e(\mathbf{x})$ can be used as a reversed confidence value, with small numbers indicating high confidence. The solution given by (7.23) exists and is unique unless the whole neighborhood is exposed to the aperture problem.

Sometimes it is useful to add a certainty weight $c(\mathbf{x} + \mathbf{\Delta x})$ to (7.22). This is most easily handled by scaling $\mathbf{A}$ and $\mathbf{\Delta b}$ accordingly.

7.5 Estimating a Parameterized Displacement Field

Like in section 6.3 we can improve robustness if the displacement field can be parameterized according to some motion model. The approach is very similar and we derive it for the eight parameter model in 2D, given by equation (6.6),

$$\begin{aligned} d_x(x,y) &= a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy, \\ d_y(x,y) &= a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2. \end{aligned} \tag{7.25}$$

We can rewrite this in matrix form similar to (6.28) and (6.29), except that we do not have an extra temporal dimension,

$$\mathbf{d} = \mathbf{S}\mathbf{p}, \tag{7.26}$$

$$\mathbf{S} = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 & x^2 & xy \\ 0 & 0 & 0 & 1 & x & y & xy & y^2 \end{pmatrix}, \tag{7.27}$$

$$\mathbf{p} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 \end{pmatrix}^T. \tag{7.28}$$

Inserting into (7.22) we obtain the weighted least squares problem

$$\sum_i w_i \|\mathbf{A}_i \mathbf{S}_i \mathbf{p} - \Delta \mathbf{b}_i\|^2, \tag{7.29}$$

where we use i to index the coordinates in a neighborhood. The solution is

$$\mathbf{p} = \left(\sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i \right)^{-1} \sum_i w_i \mathbf{S}_i^T \mathbf{A}_i^T \Delta \mathbf{b}_i. \tag{7.30}$$

We can notice that just as in section 6.3, any motion model which is linear in its parameters can be used. We also notice that like in the previous section we can compute $\mathbf{S}^T\mathbf{A}^T\mathbf{A}\mathbf{S}$ and $\mathbf{S}^T\mathbf{A}^T\Delta\mathbf{b}$ pointwise and then average these with w. In fact (7.30) reduces to (7.23) for the constant motion model.

A minor variation of the idea is to approximate the entire signal with one parametric displacement field, allowing us to compute the parameters by

$$\mathbf{p} = \left(\sum_i \mathbf{S}_i^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{S}_i \right)^{-1} \sum_i \mathbf{S}_i^T \mathbf{A}_i^T \Delta \mathbf{b}_i, \tag{7.31}$$

where the summation is over the whole signal.

7.6 Incorporating A Priori Knowledge

A principal problem with the method so far is that we assume that the local polynomials at the same coordinates in the two signals are identical except for a

displacement. Since the polynomial expansions are local models these will vary spatially, introducing errors in the constraints (7.14). For small displacements this is not too serious, but with larger displacements the problem increases. Fortunately we are not restricted to comparing two polynomials at the same coordinate. If we have a priori knowledge about the displacement field, we can compare the polynomial at $\mathbf{x}$ in the first signal to the polynomial at $\mathbf{x} + \tilde{\mathbf{d}}(\mathbf{x})$ in the second signal, where $\tilde{\mathbf{d}}(\mathbf{x})$ is the initial displacement field rounded to integer values. Then we effectively only need to estimate the relative displacement between the real value and the rounded a priori estimate, which hopefully is smaller.

This observation is included in the algorithm by replacing equations (7.12) and (7.13) by

$$\mathbf{A}(\mathbf{x}) = \frac{\mathbf{A}_1(\mathbf{x}) + \mathbf{A}_2(\tilde{\mathbf{x}})}{2}, \tag{7.32}$$

$$\Delta\mathbf{b}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2(\tilde{\mathbf{x}}) - \mathbf{b}_1(\mathbf{x})) + \mathbf{A}(\mathbf{x})\tilde{\mathbf{d}}(\mathbf{x}), \tag{7.33}$$

where

$$\tilde{\mathbf{x}} = \mathbf{x} + \tilde{\mathbf{d}}(\mathbf{x}). \tag{7.34}$$

The first two terms in $\Delta\mathbf{b}$ are involved in computing the remaining displacement while the last terms adds back the rounded a priori displacement. We can see that for $\tilde{\mathbf{d}}$ identically zero, these equations revert to (7.12) and (7.13), as would be expected.

The displacement estimation algorithm derived in the last three sections is illustrated with a block diagram in figure 7.8. Input are the quadratic polynomial expansion coefficients for the two signals, $\mathbf{A}_1, \mathbf{b}_1, \mathbf{A}_2, \mathbf{b}_2$, and an a priori displacement field $\mathbf{d}_{\text{in}}$. Output is the estimated displacement field $\mathbf{d}_{\text{out}}$.

7.7 Iterative and Multi-scale Displacement Estimation

A consequence of the inclusion of an a priori displacement field in the algorithm is that we can close the loop and iterate. A better a priori estimate means a smaller relative displacement, which in turn improves the chances for a good displacement estimate. We consider two different approaches, iterative displacement estimation and multi-scale displacement estimation.

7.7.1 Iterative Displacement Estimation

The simplest solution is shown in figure 7.9, where the displacement estimation is iterated three times. The output displacement of one step is used as a priori displacement in the next step. The a priori displacement field $\mathbf{d}_{\text{in}}$ at the first step would usually be set to zero, unless actual knowledge about the displacement field is available. The same polynomial expansion coefficients are used in all steps and need only be computed once. It is of course possible to use any fixed number of iterations or to iterate until the displacement estimates have converged.

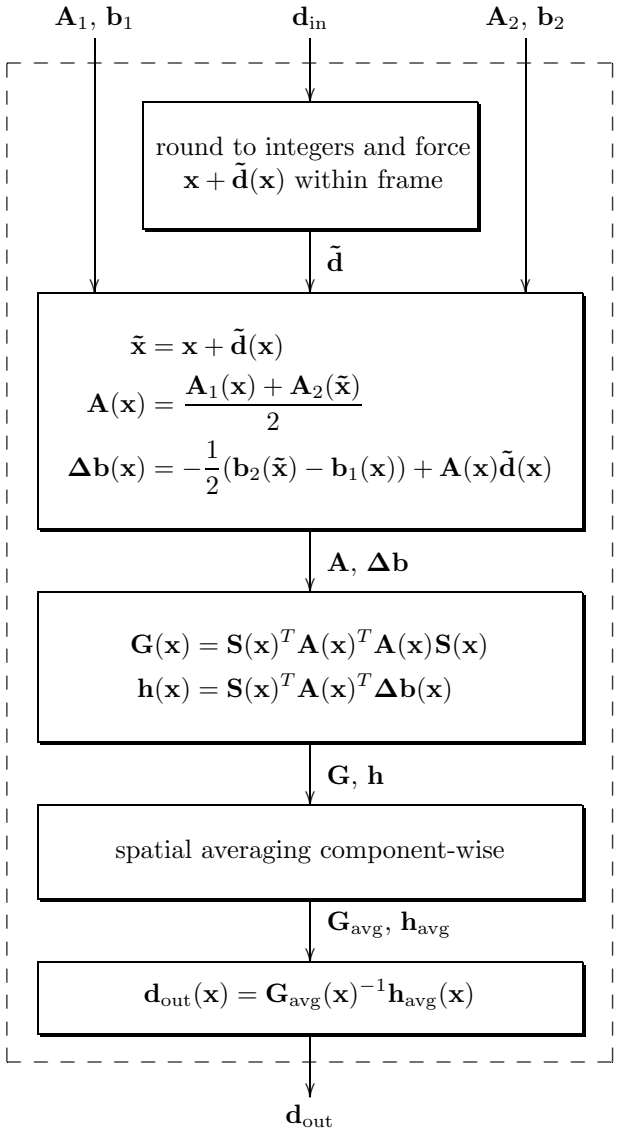
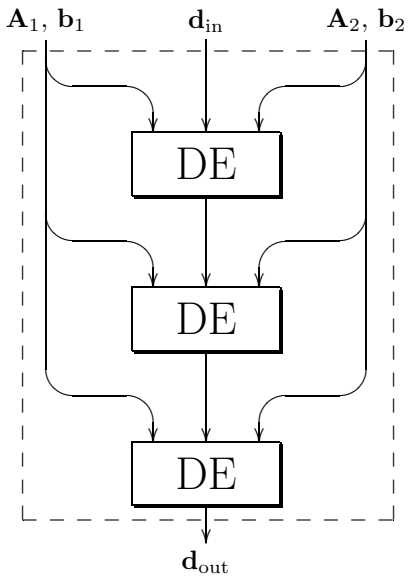


Figure 7.8: Block diagram of the displacement estimation algorithm.



DE = Displacement Estimation block from figure 7.8

Figure 7.9: Block diagram of the iterative displacement estimation algorithm.

The weak spot of this approach is in the first iteration. If the displacements (relative the a priori displacements) are too large, the output displacements cannot be expected to be improvements and iterating will be meaningless.

7.7.2 Multi-scale Displacement Estimation

The problem of too large displacements can be reduced by doing the analysis at a coarser scale. This means that we use a larger applicability for the polynomial expansion and/or lowpass filter the signal first, as discussed in section 4.5. The effect is that the estimation algorithm can handle larger displacements but at the same time the accuracy decreases.

This observation points to a multi-scale approach. Start at a coarse scale to get a rough but reasonable displacement estimate and propagate this through finer scales to obtain increasingly more accurate estimates. Figure 7.10 shows a diagram for a three-scale displacement estimation algorithm. To reduce computations the signals f_1 and f_2 are both lowpass filtered and subsampled between scales but the algorithm would work with any multi-scale polynomial expansion scheme. If the signal is subsampled it is necessary to do an upsampling of the estimated displacement fields between the scales, including rescaling the values appropriately. As before we set the a priori displacement $\mathbf{d}_{\text{in}}$ at the coarsest scale to zero, unless actual knowledge about the displacement field is available.

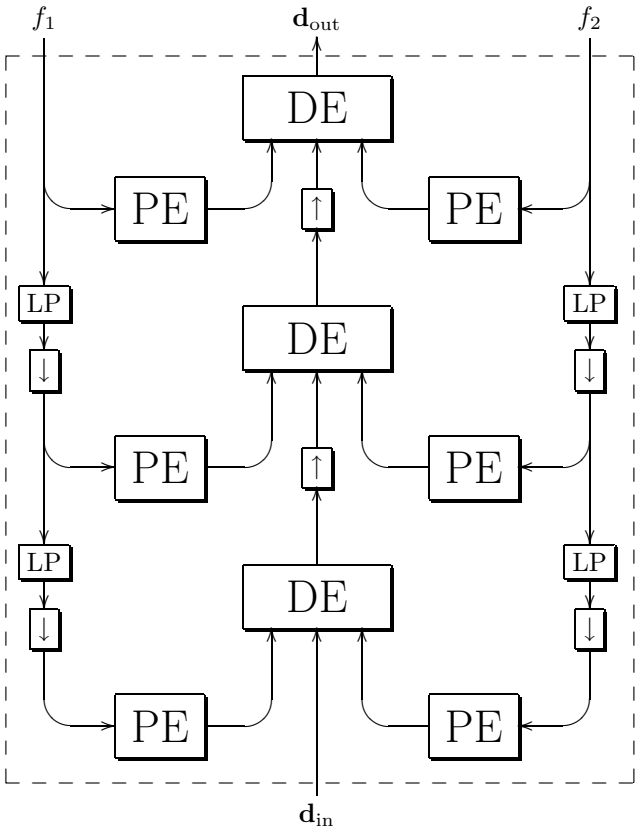
Compared to the iterative displacement estimation algorithm, this approach requires new polynomial expansion coefficients to be computed for each scale. As we can see in the next section, however, this has only minor effects on the computational complexity, if we do subsample. It is also conceivable to combine both approaches and iterate multiple times at each scale but that is probably not an efficient solution, except possibly at the coarsest scale.

7.8 Computational Complexity

The computational complexity of the displacement estimation algorithms is dominated by two steps; the polynomial expansion and the spatial averaging of the $\mathbf{G}$ and $\mathbf{h}$ matrices in figure 7.8.

The cost of polynomial expansion is discussed in section 4.4. For displacement estimation we need two polynomial expansions, one for each signal. If we are using the multi-scale algorithm we also need to recompute the polynomial expansions for each scale. However, if subsampling is employed the number of pixels at each scale typically decreases by a factor of two per dimension, meaning that the cost for all remaining scales is only a fraction of the cost for the first scale.

The averaging operation can be assumed to be implemented by separable filtering. Letting n be the length of the separable filters and d the dimensionality, we have dn operations per pixel and component in G and h. The number of components obviously depends on the number of parameters in the motion model. Letting this be m we have an upper limit of $\frac{m(m+1)}{2}$ independent components in $\mathbf{G}$ since this is a symmetric $m \times m$ matrix and we have m components in the vector $\mathbf{h}$. In practice there are often a few more duplicate components in $\mathbf{G}$, however. In



DE = Displacement Estimation block from figure 7.8
PE = Polynomial Expansion
LP = LowPass filter
↓ = Downsampling
↑ = Upsampling

Figure 7.10: Block diagram of the multi-scale displacement estimation algorithm.

2D we have a total of 5 components in $\mathbf{G}$ and $\mathbf{h}$ for the constant motion model[1], 24 for the affine motion model, and 39 for the eight-parameter motion model. For the iterative algorithm we also need to multiply with the number of iterations and for the multi-scale algorithm with the number of scales, unless subsampling is used, which makes the additional cost small. To summarize, if we let k be the effective number of iterations, the cost for this step is in the order of $\frac{dnm^2k}{2}$ operations per pixel.

7.9 Evaluation

To verify that the algorithms are basically sound and to explore some of their characteristics we have made a very simple experiment, involving global translations only. We have also tested the algorithms on the Yosemite and the diverging tree sequences, as well as on some real image sequences.

7.9.1 Global Translation Experiment

The setup for the first experiment is that the first image is the 512×512 Lena image, also used in section 3.9. The second image is obtained by translating the first image some distance, using cubic interpolation. To avoid any complications related to borders or missing data, the global displacement is estimated by summing over the central 256×256 pixels in equation (7.31), using the constant motion model.

The algorithms have been tested with translations ranging from 0.2 pixels and upwards in steps of 0.2 pixels and in 16 different directions. Figure 7.11(a) shows a truncated plot of the translation vectors.

All polynomial expansions have been computed with Gaussian applicability with $\sigma = 0.15(N-1)$, where N is the spatial size of the filters. The initial a priori displacements have been set to zero in all cases.

We refer to the different displacement estimation algorithms as "basic" for figure 7.8, "iterative" for figure 7.9, and "multi-scale" for figure 7.10. In the latter case, however, we use no subsampling or lowpass filtering in the polynomial expansion structure. This makes the results more directly comparable.

Figure 7.11(b) shows a plot of the estimated displacements when using the basic algorithm and filter size $N = 17$. The ideal result would have looked exactly like figure 7.11(a) and what we can see is that the estimates are reasonable up to a distance of about 6 pixels, after which the algorithm can no longer follow the displacements. What happens is that more and more of the constraints (7.14) become essentially random and together tend to favor a small displacement estimate. Figure 7.11(c) shows a close-up of the results from one of the directions together with the correct results and we can see that the angular errors are relatively small but that the distances become increasingly overestimated. This holds up to the point where the algorithm can no longer follow the displacements. However, due to the previous overestimation, the absolute errors happen to improve for a while when the estimates start to become smaller.

[1] 6 if we also want to compute the confidence measure (7.24).

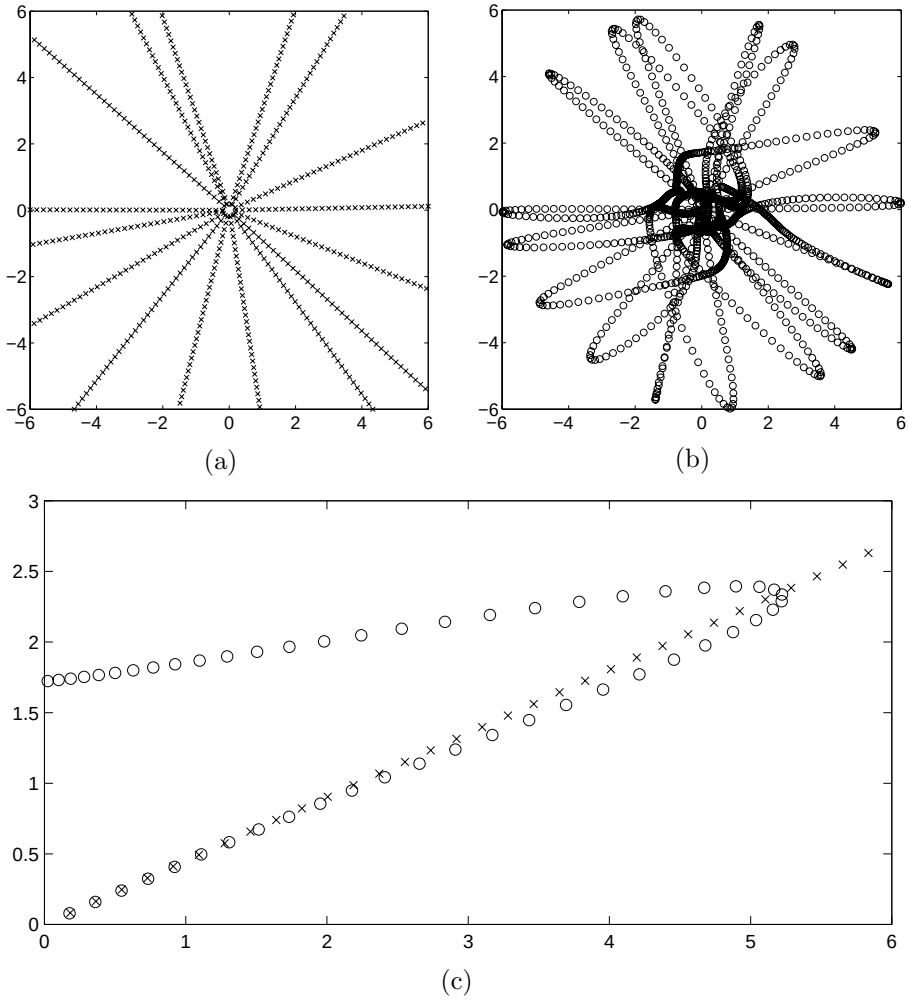


Figure 7.11: Correct and estimated displacement vectors. Basic algorithm with $N = 17$.

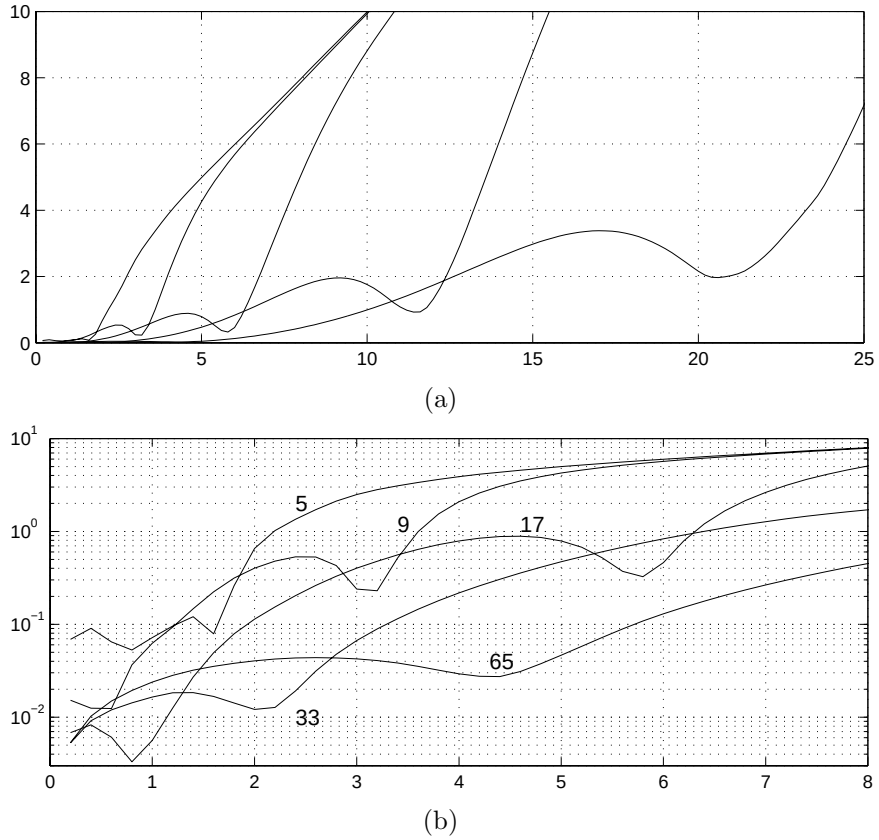


Figure 7.12: Average absolute displacement errors, plotted against distance. Basic algorithm.

Figure 7.12(a) shows the absolute errors, averaged over the 16 directions and plotted against the distance. The five curves are for the filter sizes 5, 9, 17, 33, and 65 respectively. As expected we can recover larger displacements at coarser scales. Figure 7.12(b) shows a close-up of the curves in a logarithmic plot and we can see that for small displacements we can achieve errors in the order of 0.01 pixels, except for $N = 5$ where the filters simply are too small.

Similar plots for the iterative algorithm can be found in figure 7.13(a),(b) for $N = 9$ and $N = 17$. The six curves in each plot show the results for the first to the sixth iteration. As can be seen in the logarithmic curves in figure 7.13(c),(d) we can achieve very small errors all the way up to a distance of about 4.5 and 10 pixels respectively but beyond that there is essentially no benefit in iterating.

Figure 7.14 shows the corresponding results for the multi-scale algorithm, using filter sizes 65, 33, 17, 9, and 5 in sequence.

It should be mentioned that this experiment is extremely favorable for the

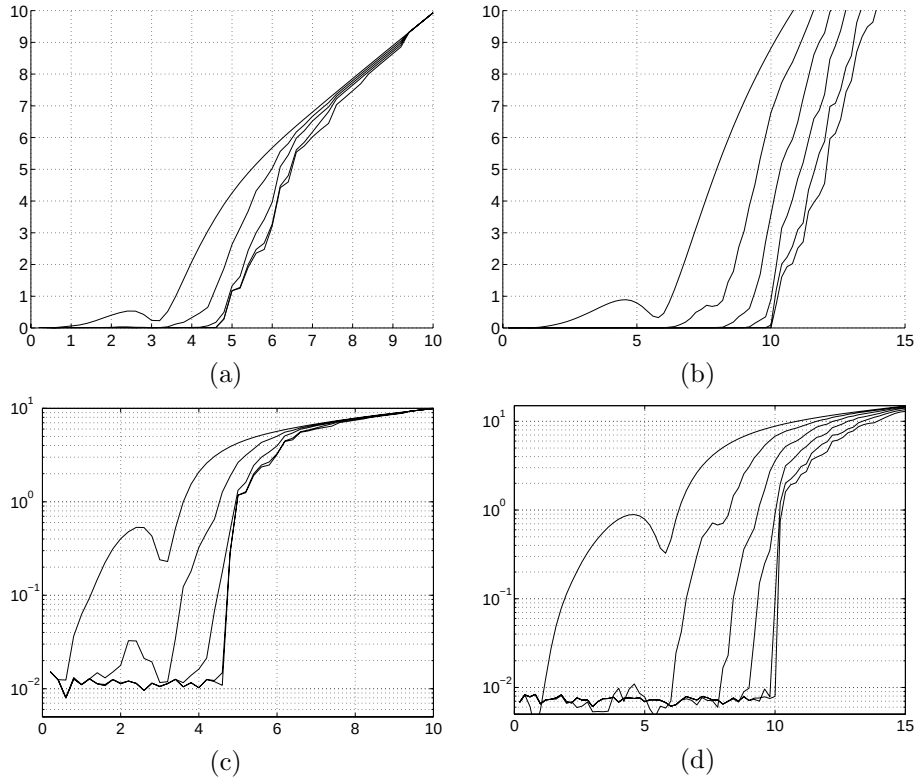


Figure 7.13: Average absolute displacement errors, plotted against distance. Iterative algorithm, $N = 9$ (a),(c) and $N = 17$ (b),(d).

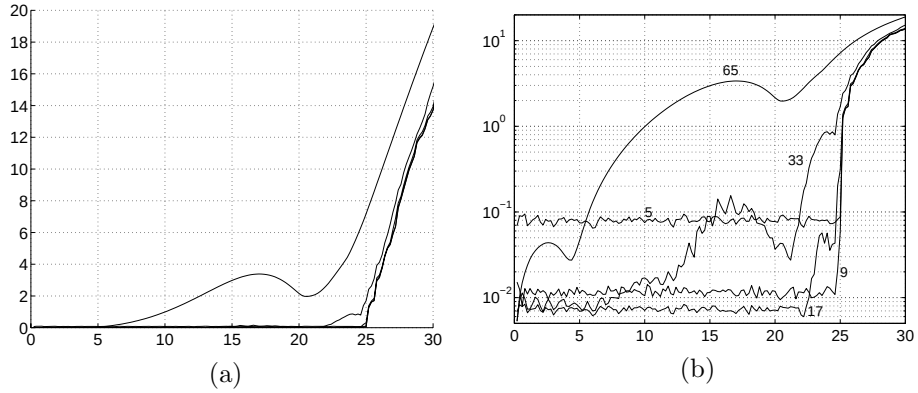


Figure 7.14: Average absolute displacement errors, plotted against distance. Multi-scale algorithm.

Technique	Average error	Standard deviation	Density
Constant, 1 iteration	3.94°	4.24°	100%
	3.17°	2.58°	70%
	2.10°	1.51°	30%
Constant, 3 iterations	2.64°	2.60°	100%
	2.18°	1.84°	70%
	1.64°	1.30°	30%
Affine, 1 iteration	4.20°	6.79°	100%
Affine, 3 iterations	2.36°	3.69°	100%

Table 7.1: Results for the Yosemite sequence using constant and affine motion models.

algorithms in that only one of the assumptions discussed in section 7.2.2 is broken[2] and we know that we can collect information over a large region. We can in particular see that we obtain high accuracy also for the coarsest scales, which cannot be expected when the translation stops being global. As a consequence the multi-scale algorithm does not come to its right because the errors would have become smaller by iterating within the $N = 33$ or $N = 17$ scale instead of continuing to finer scales[3].

7.9.2 Results for Yosemite and Diverging Tree

The Yosemite and diverging tree sequences were presented in section 6.6. We have estimated the displacement from the center frame and the following frame. This is done for the basic and the iterative algorithms using a 39×39 Gaussian weighting function (w in equation (7.29)) with standard deviation 6. The polynomial expansion is done with an 11×11 Gaussian applicability with standard deviation 1.5 for the Yosemite sequence and a 9×9 Gaussian with standard deviation 1.2 for the diverging tree sequence. In order to avoid large errors near the borders, the polynomial expansions have been computed with the separable normalized convolution method. Additionally pixels close to the borders have been given a reduced certainty (see section 7.4) because the expansion coefficients still can be assumed to be less reliable there.

The results are shown in table 7.1 for the Yosemite sequence and in table 7.2 for the diverging tree sequence. Comparison with other methods can be found in section 6.6. The results for partial coverage have been computed from the most reliable estimates according to the confidence measure (7.24).

We can see that these algoritms do not at all perform as well as the tensor-based algorithms in chapter 6. This is not very surprising since these algorithms only use two frames while the tensor-based methods take advantage of the spatio-temporal consistency over several frames. Still the Yosemite results are relatively

[2]The polynomials are local but the displacement is still a global translation.
[3]The $N = 5$ scale is probably always too fine, however.

Technique	Average error	Standard deviation	Density
Constant, 1 iteration	3.59°	2.60°	100%
	3.57°	2.42°	70%
	4.02°	2.19°	30%
Constant, 3 iterations	3.68°	3.27°	100%
	3.46°	2.31°	70%
	3.93°	2.23°	30%
Affine, 1 iteration	3.61°	5.33°	100%
Affine, 3 iterations	3.13°	5.59°	100%

Table 7.2: Results for the diverging tree sequence using constant and affine motion models.

good.

It is worth noticing that the confidence measure works reasonably well for the Yosemite sequence but not at all for the diverging tree sequence, which was also the case for the tensor based methods, see section 6.6.3. We can also see that the results are not always improved by iterating.

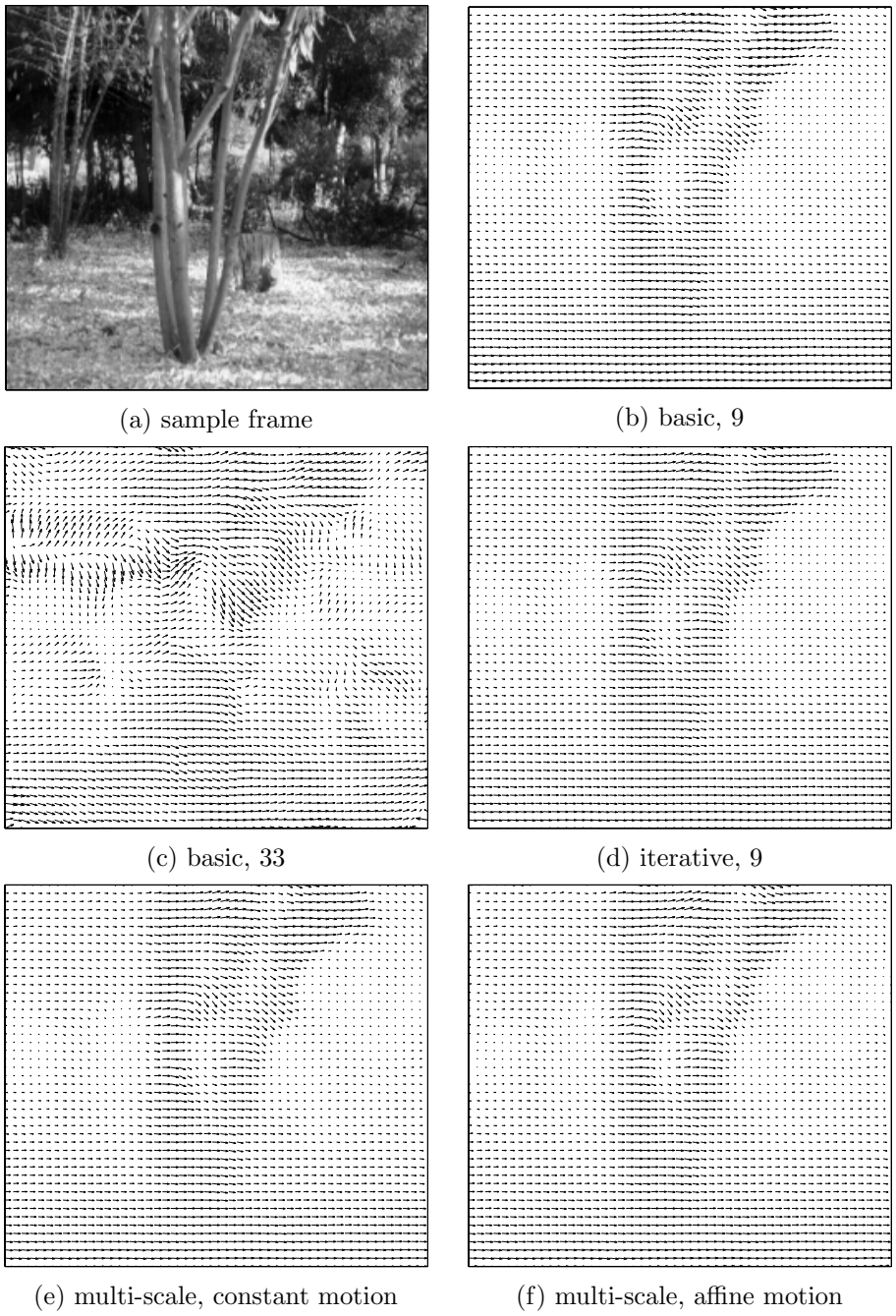
7.9.3 Results for Real Image Sequences

As in section 6.6.4 we have tested the displacement estimation algorithms on three real image sequences and the results can be found in figures 7.15–7.17. Each figure contains five displacement fields. Common for all of them is that w is a 39×39 Gaussian with standard deviation 6 and that the polynomial expansion is done with an $N \times N$ Gaussian applicability with standard deviation $0.15(N-1)$. As before the pixels close to the borders have been given a reduced certainty. The differences are:

(b) Basic algorithm, $N = 9$, constant motion model.
(c) Basic algorithm, $N = 33$, constant motion model.
(d) Iterative algorithm, $N = 9$, 3 iterations, constant motion model.
(e) Multi-scale algorithm, $N = 33$, 17, 9, constant motion model.
(f) Multi-scale algorithm, $N = 33$, 17, 9, affine motion model.

7.10 Moving Object Detection

Returning to the image sequence in figure 7.1, we are interested in detecting moving objects. In this case we have two cars which are moving slowly through the crossing. Unfortunately this is difficult to see from the displacement fields in figure 7.2, estimated by the multi-scale algorithm over three scales and with the constant motion model, since they are completely dominated by camera ego-motion, mostly due to vibrations.

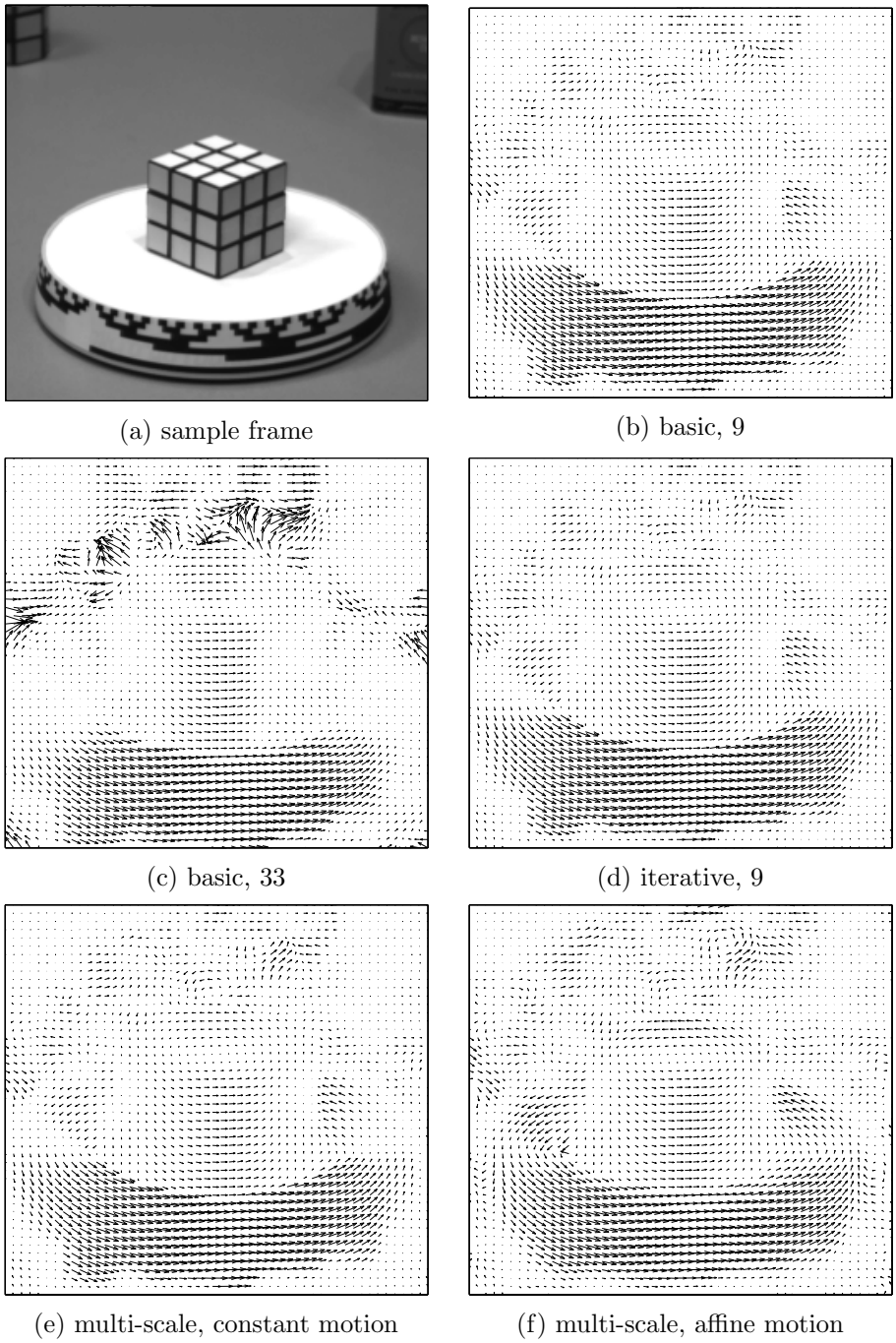


(a) sample frame (b) basic, 9

(c) basic, 33 (d) iterative, 9

(e) multi-scale, constant motion (f) multi-scale, affine motion

Figure 7.15: SRI Trees sequence. Sample frame and estimated displacement fields (subsampled).

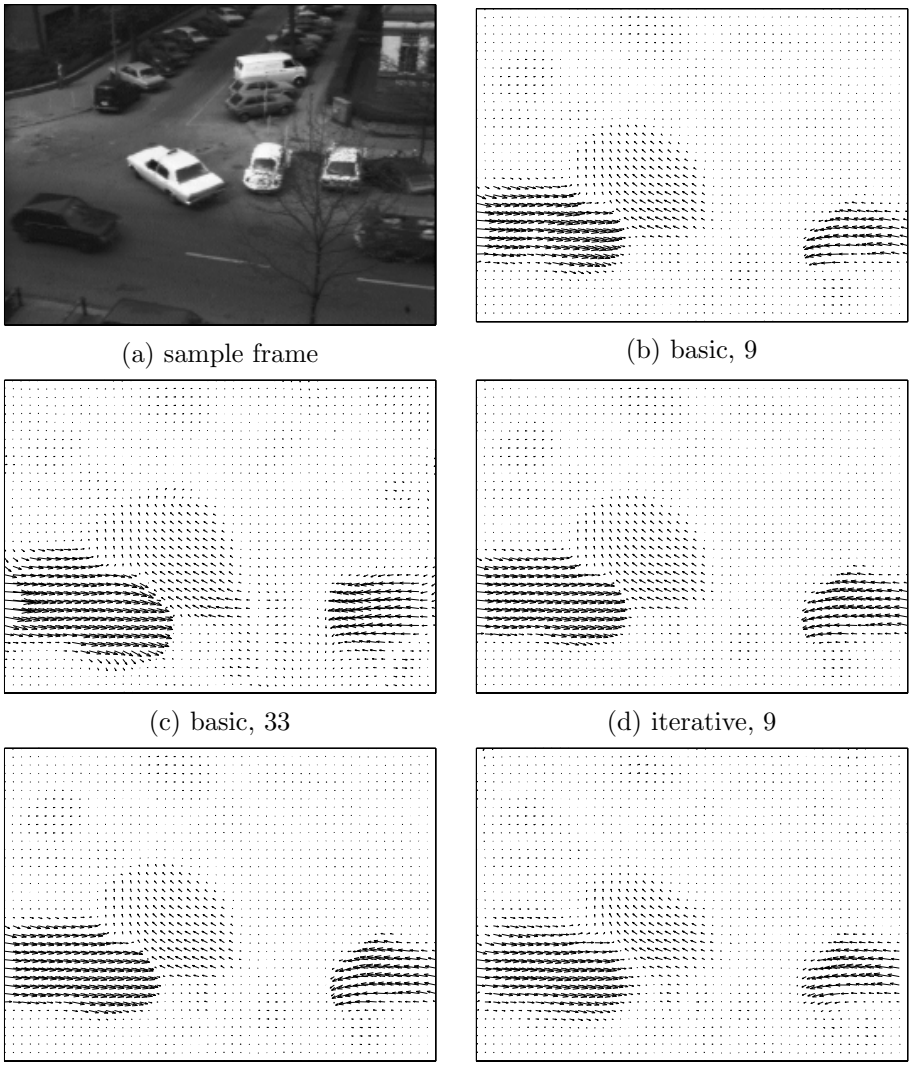


(a) sample frame (b) basic, 9

(c) basic, 33 (d) iterative, 9

(e) multi-scale, constant motion (f) multi-scale, affine motion

Figure 7.16: Rubik Cube sequence. Sample frame and estimated displacement fields (subsampled).

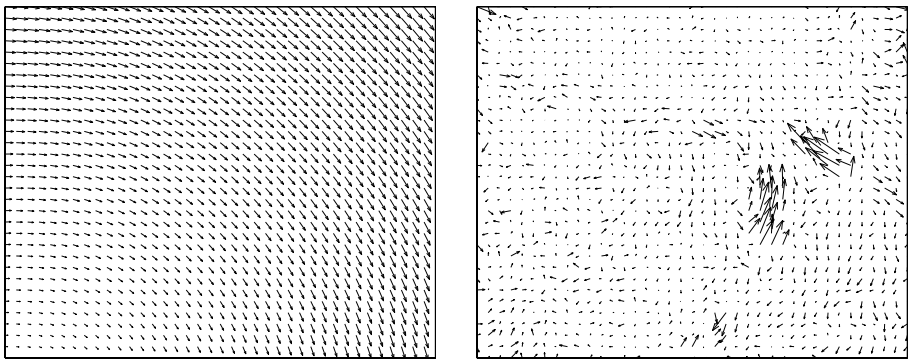


(a) sample frame (b) basic, 9

(c) basic, 33 (d) iterative, 9

(e) multi-scale, constant motion (f) multi-scale, affine motion

Figure 7.17: Hamburg Taxi sequence. Sample frame and estimated displacement fields (subsampled).



(a) background displacement (b) residual displacement

Figure 7.18: Background and residual displacement fields corresponding to figure 7.2(a). The residual field is magnified by a factor 10.

7.10.1 The Plane+Parallax Decomposition

To solve the problem we use the plane+parallax approach [48, 67, 68, 76, 78]. The idea is that the background can be approximated by a reference plane, the displacement field of which can be fit to the eight-parameter motion model (7.25). After subtracting this we obtain a residual parallax displacement field where moving objects turn up and can be detected. Unfortunately also structures not lying in the reference plane cause a residual displacement, so further processing is required to distinguish these. In principle it is possible to use the fact that the parallax induced by stationary objects constitutes an epipolar field [47] but it is probably more robust and efficient to sort out potential moving objects by using other cues such as size or temporal coherence.

7.10.2 Estimating Background Displacement

One straightforward way to compute the background displacement would be to fit the estimated displacement field globally to the eight-parameter motion model (7.25). This would be unnecessarily indirect, however, since equation (7.31) already tells us how to compute a global parametric field directly from the primary constraints. It is worth noting that this is much less expensive than estimating parametric fields locally, since only a single summation over the whole image is involved.

Figure 7.18 shows the estimated background displacement field corresponding to figure 7.2(a) and the residual displacement field. We can now easily detect the moving cars, although the field still is somewhat noisy.

It is implicitly assumed that the moving objects cover a sufficiently small area compared to the background, so that their contribution to the sum in (7.31) does not significantly affect the estimated parameters.

7.10.3 Reducing Noise

A closer investigation of the residual displacement field shows that most of the noise originates from areas without significant structures or with very low contrast. A special case are areas where the aperture problem is apparent and there is noise in the parallel displacement component.

A solution to this problem is to "force" the background field onto uncertain estimates. We do this by adding a regularization term to equation (7.22), i.e. minimize

$$\mu\|\mathbf{d}(\mathbf{x}) - \mathbf{d}_0(\mathbf{x})\|^2 + \sum_{\mathbf{\Delta x}\in I} w(\mathbf{\Delta x})\|\mathbf{A}(\mathbf{x}+\mathbf{\Delta x})\mathbf{d}(\mathbf{x}) - \mathbf{\Delta b}(\mathbf{x}+\mathbf{\Delta x})\|^2, \quad (7.35)$$

where $\mathbf{d}_0$ is the previously estimated background displacement field and μ is a constant. The idea is that the regularization term will have little effect when the displacement is well constrained by the sum in the expression but be significant when it is not. This works well with the aperture problem, where the normal component is well constrained but not the parallel component. The solution to (7.35) is given by

$$\mathbf{d}(\mathbf{x}) = \left(\mu\mathbf{I} + \sum w\mathbf{A}^T\mathbf{A}\right)^{-1}\left(\mu\mathbf{d}_0(\mathbf{x}) + \sum w\mathbf{A}^T\mathbf{\Delta b}\right), \quad (7.36)$$

where we again have dropped some indexing to improve the readability.

Figure 7.19 shows a block diagram for the revised basic displacement estimation algorithm. As before this can be combined with figure 7.9 or figure 7.10 to obtain iterative or multi-scale versions of the algorithm.

How to choose μ is an open question. The alternative we have tested is to set μ to the average of half the trace of $\mathbf{G}_{\text{avg}}$ (using the notation from figure 7.19), computed over the whole image. This reduces the noise significantly but as a drawback it also tends to reduce the size of the real residuals. For the purpose of detecting motion this is probably a reasonable trade-off.[4] Figure 7.20 shows the total and the residual displacement fields from figure 7.2(a) and figure 7.18(b) recomputed by this revised algorithm.

7.11 Discussion

Although the derivation of the primary constraint (7.14) may look odd and involves a number of more or less likely-to-hold assumptions, it turns out that, in an important special case, the constraint itself is not strange at all. To see this we need to restrict ourselves to Gaussian applicabilities for the polynomial expansion, in which case section 4.8.1 tells us that $\mathbf{b}$ and $\mathbf{A}$ contain Gaussian derivatives of the first and second order.

Now consider the second order brightness change constraint equation,

$$\begin{pmatrix} I_{xx} & I_{yx} \\ I_{xy} & I_{yy} \end{pmatrix}\begin{pmatrix} d_x \\ d_y \end{pmatrix} = -\begin{pmatrix} I_{tx} \\ I_{ty} \end{pmatrix}. \quad (7.37)$$

[4] Once the moving objects have been detected, their displacements can be recomputed without regularization, if a more correct displacement estimate is needed.

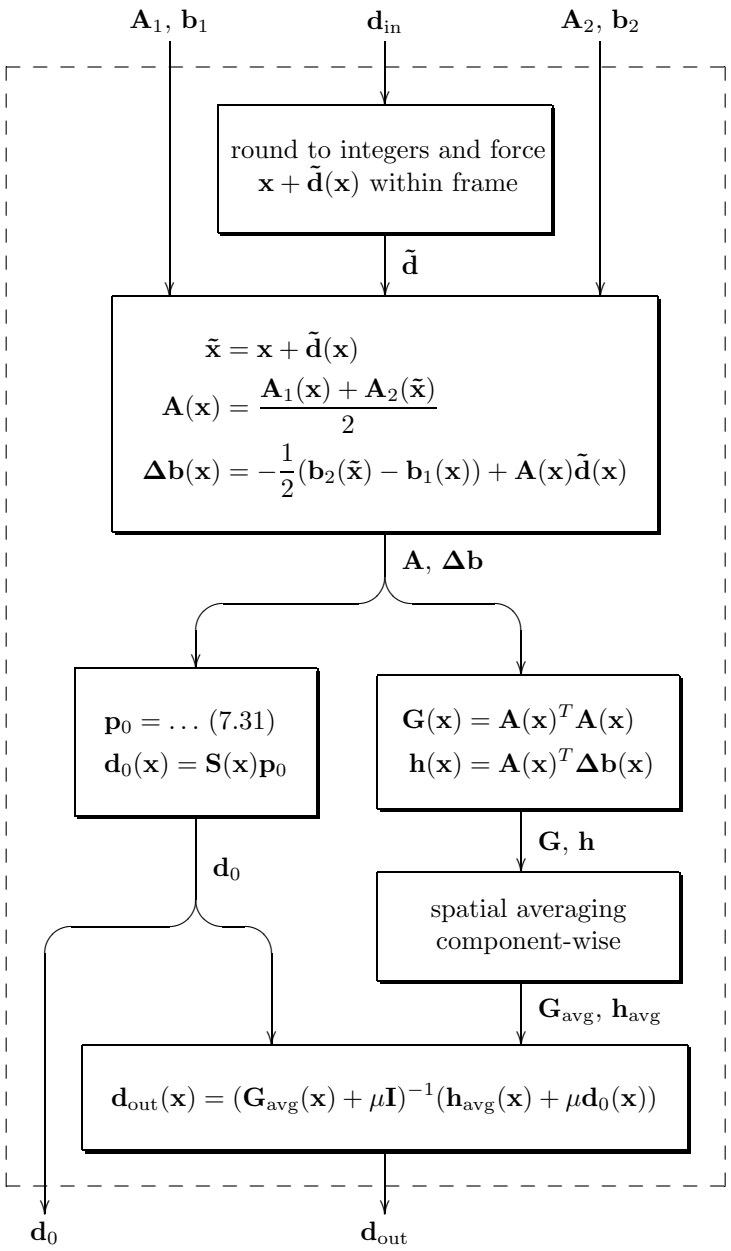
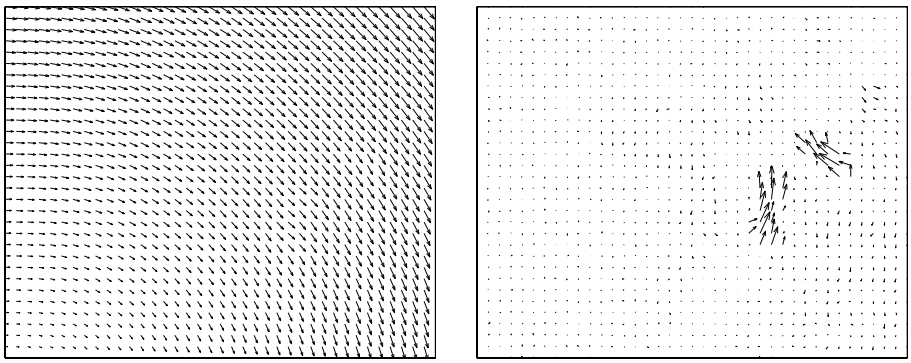


Figure 7.19: Block diagram of the revised basic displacement estimation algorithm.



(a) total displacement (b) residual displacement

Figure 7.20: Total and residual displacement fields computed by the revised algorithm. The residual field is magnified by a factor 10.

If we assume that the left hand matrix is computed with a temporal smoothing over two frames, this is the same as $\mathbf{A}_1+\mathbf{A}_2$. The right hand side equals $-(\mathbf{b}_2-\mathbf{b}_1)$ if we let the temporal differentiation be approximated by a subtraction. All this assumes, of course, that the remaining derivatives all are of the Gaussian variety.

Hence the two constraints coincide in this special case, for good and for bad. Good because it adds to the credibility of the method. Bad because it makes it less novel than we first thought. Still we believe that the alternative derivation is a valuable insight by itself and that it makes the important observation about inclusion of a priori displacement estimates in section 7.6 more natural and easier to find. Besides there is a difference if the applicability is not Gaussian or if we do not have full certainty.

A potential improvement, unique to the polynomial approach, would be to also make use of the constraint (7.5), in addition to (7.4). So far attempts at this have not been successful, however. It is also conceivable to take the dissimilarity between $\mathbf{A}_1$ and $\mathbf{A}_2$ into account to modify the weight of the constraint.

An interesting generalization is to consider not only a translation of a polynomial, as in section 7.2, but also include e.g. rotation. With $\mathbf{R}$ a rotation matrix, the corresponding derivation becomes

$$\begin{aligned}
f_1(\mathbf{x}) &= \mathbf{x}^T\mathbf{A}_1\mathbf{x} + \mathbf{b}_1^T\mathbf{x} + c_1, \\
f_2(\mathbf{x}) &= f_1(\mathbf{R}\mathbf{x} - \mathbf{d}) \\
&= (\mathbf{R}\mathbf{x} - \mathbf{d})^T\mathbf{A}_1(\mathbf{R}\mathbf{x} - \mathbf{d}) + \mathbf{b}_1^T(\mathbf{R}\mathbf{x} - \mathbf{d}) + c_1 \\
&= \mathbf{x}^T\mathbf{R}^T\mathbf{A}_1\mathbf{R}\mathbf{x} - 2\mathbf{d}^T\mathbf{A}_1\mathbf{R}\mathbf{x} + \mathbf{d}^T\mathbf{A}_1\mathbf{d} + (\mathbf{R}^T\mathbf{b}_1)^T\mathbf{x} - \mathbf{b}_1^T\mathbf{d} + c_1 \\
&= \mathbf{x}^T(\mathbf{R}^T\mathbf{A}_1\mathbf{R})\mathbf{x} + (\mathbf{R}^T\mathbf{b}_1 - 2\mathbf{R}^T\mathbf{A}_1\mathbf{d})^T\mathbf{x} + \mathbf{d}^T\mathbf{A}_1\mathbf{d} - \mathbf{b}_1^T\mathbf{d} + c_1 \\
&= \mathbf{x}^T\mathbf{A}_2\mathbf{x} + \mathbf{b}_2^T\mathbf{x} + c_2
\end{aligned} \tag{7.38}$$

and by equating coefficients we obtain

$$\mathbf{A}_2 = \mathbf{R}^T \mathbf{A}_1 \mathbf{R}, \tag{7.39}$$

$$\mathbf{b}_2 = \mathbf{R}^T (\mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d}). \tag{7.40}$$

This approach has not been explored further but may be useful if the motion is expected to involve substantial rotations. Instead of rotation we can of course consider e.g. scaling or a full affine transformation.

Finally there may be some potential for improvements by investigating the systematic overestimation of the displacement noticed in figure 7.11(c) and if possible compensate for it.

Chapter 8

Analysis of Interlaced Signals

8.1 Introduction

Almost all implementations of image analysis algorithms require data discretized on a rectangular grid. This is natural since that format is both convenient for theoretical analysis and well adapted to commonly available hardware. If the data for some reason comes in another, less regular, format, the first processing step is usually to resample it to a rectangular grid. While this undeniably is an effective solution, it does have a computational cost and may cause a loss of information. These problems can be avoided if we can design at least the first level of processing to work directly on the less regular data.

In this chapter we only look at one special but important case, namely interlaced video signals. Interlacing was introduced into the standard television formats to reduce the bandwidth compared to progressive sampling, at an equivalent level of flicker perception [18]. For computer vision applications progressive sampling would be more convenient but video cameras, at least at the consumer-level, do produce interlaced video sequences, so we need methods to handle them.

The fact that all algorithms in this thesis are based (directly or indirectly) on normalized convolution makes it very straightforward to adapt them to interlaced signals. In particular, once we have established how to compute polynomial expansion, the orientation and motion estimation algorithms in chapters 5–7 can be used right away.

8.2 The Geometry of Interlaced Signals

The basic idea behind interlacing is to trade a halving of vertical resolution for a doubling of temporal resolution, in such a way that the full vertical detail is obtained over two time samples. Letting x be the horizontal direction and y the vertical direction, figure 8.1 shows the sampling pattern in the ty-plane. This

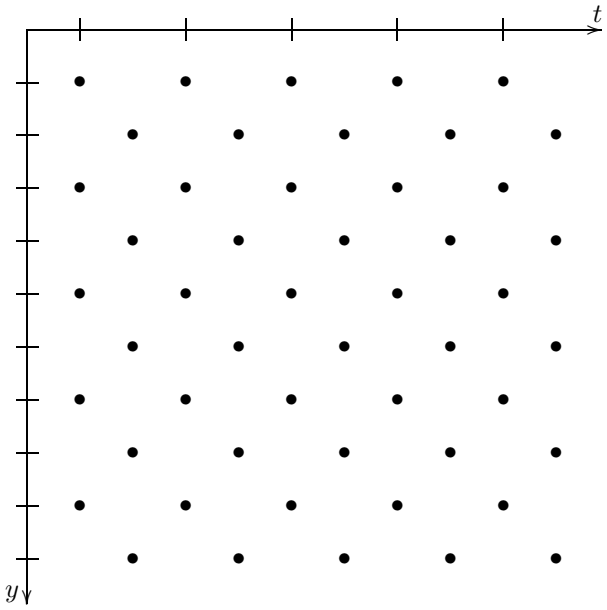


Figure 8.1: Sampling pattern for interlaced video signals, ty-plane.

pattern is repeated along the x-axis.

When the video stream is digitized the sampling pattern is collapsed so that two time instants, called fields, are merged into one frame, as shown in figure 8.2. Thus the odd lines belong to time instant t while the even lines belong to time instant $t + \frac{1}{2}$.[1] An example frame is shown in figure 8.3 together with a magnified area that shows the kind of artefacts produced if the interlaced geometry is ignored[2].

To conclude, an interlaced video sequence, as it appears to the vision algorithms, is a sequence of frames sampled according to figure 8.2 but where the actual geometry is given by figure 8.1.

8.3 Normalized Convolution

Normalized convolution has an extremely straightforward adaptation to interlaced signals. Just repack the signal according to figure 8.4, which is the sampling pattern in figure 8.1 padded with zeros to a regular grid, and set up a corresponding certainty which is one at the sample points and zero otherwise[3]. Then normalized

[1] Or the other way round, depending on where you start counting.

[2] This is the same image as in figure 7.1(a). In chapter 7 the interlacing was crudely eliminated by subsampling.

[3] If we already have certainty values for the interlaced samples we just pad those with zeros. Since the signal values at points with certainty zero do not matter, we can construct the repacked signal volume simply by doubling each frame, if this is more convenient, or leave the unused

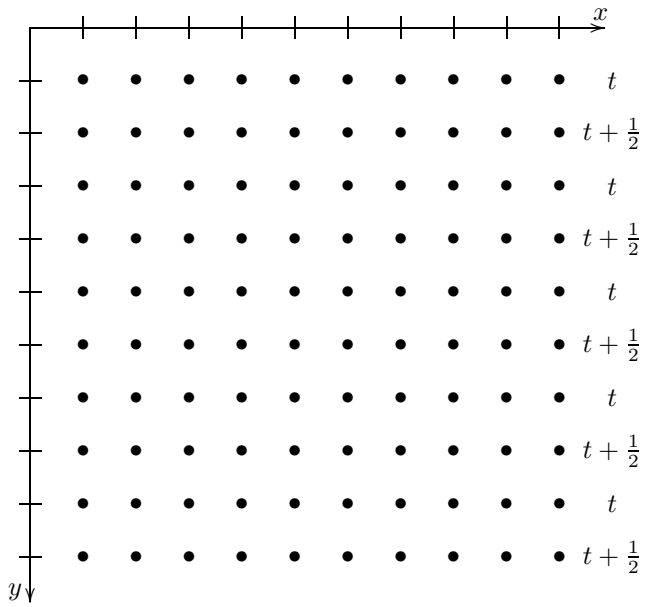


Figure 8.2: Two fields merged into one frame, xy-plane.

(a) (b)

Figure 8.3: Sample interlaced frame and zoom-in.

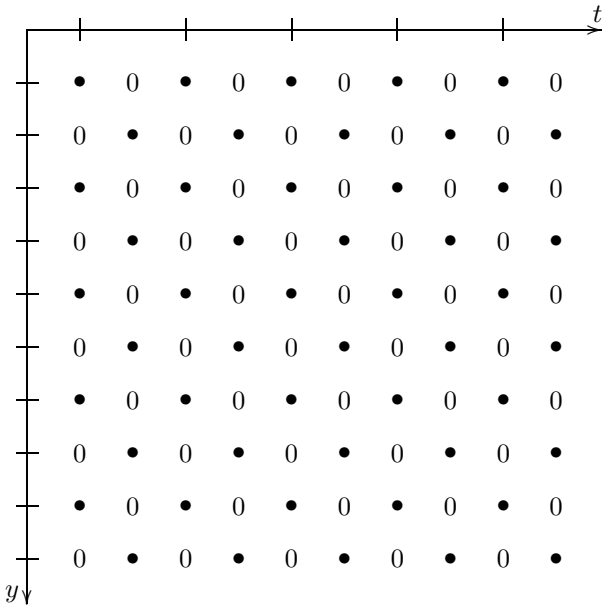


Figure 8.4: Sampling pattern in figure 8.1 padded with zeros to a regular grid.

convolution can be used without further modifications. In the case of 2D normalized convolution in the xy-plane, i.e for a single field, there is not even a need to repack the signal. It suffices to use a certainty which is zero for every second row, depending on which field should be used. Another plausible way to avoid the repacking is to create a dedicated normalized convolution implementation which addresses the samples in the original sequence properly.

Assuming that we have a binary certainty, only caused by the interlaced sampling pattern, and that we are far from the borders or ignore them, then the certainty values for a neighborhood only appear in two different ways. Either the neighborhood is centered at a sample point or it is centered at a non-sample point. This means that we can use equation (3.23),

$$\tilde{\mathbf{B}}^* = (\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c\mathbf{B})^{-1}\mathbf{B}^*\mathbf{W}_a\mathbf{W}_c \tag{8.1}$$

to compute two sets of equivalent correlation kernels[4]. The construction guarantees, as would be expected, that only real samples are accessed by the correlation kernels. If desired these can also be remapped to work directly on the original interlaced signal.

entries uninitialized.

[4]One for use at the sample points and one for use at the non-sample points.

8.4 Adaptation of Convolution Kernels

Assume that we have a set of convolution kernels designed for use on regularly sampled signals, which we need to adapt for use on interlaced signals. As in section 3.8 we convert these into corresponding correlation kernels and collect them into a matrix $\mathbf{H}$. Equation (3.33),

$$\mathbf{B} = \mathbf{W}_a^{-1}\mathbf{H}(\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{H})^{-1}, \tag{8.2}$$

shows how to compute corresponding basis functions. These can be entered into (8.1) to obtain correlation kernels adapted for interlaced signals. After simplification this becomes

$$\tilde{\mathbf{B}} = \mathbf{W}_c\mathbf{H}(\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{W}_c\mathbf{H})^{-1}\mathbf{H}^*\mathbf{W}_a^{-1}\mathbf{H}. \tag{8.3}$$

As before we have to consider two different $\mathbf{W}_c$ and thus obtain two sets of adapted filters. The practical considerations in section 3.8.2 still apply and there are reasons to be extra careful in this case, particularly with short or sparse filters. The first factor $\mathbf{W}_c$ in (8.3) effectively masks out the coefficients in the original filter kernels corresponding to the missing samples, which may under unfortunate circumstances completely destroy the desired properties of the filters.

8.5 Polynomial Expansion

Polynomial expansion is a special case of normalized convolution, so it is clear from section 8.3 that we can compute it for interlaced signals. The interesting question is to what extent the performance optimizations in section 4.3 for quadratic polynomial expansion can still be used.

To begin with we need to take the two possible certainty masks discussed in section 8.3 into account. For a 5×5 neighborhood in the ty-plane we have the certainty mask

$$\mathbf{c}_{\text{on}} = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{pmatrix} \tag{8.4}$$

centered at a sample point and

$$\mathbf{c}_{\text{off}} = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \tag{8.5}$$

centered at a non-sample point. These generalize in the obvious way to neighborhoods of different sizes and by duplication into the x direction.

If we redo the derivation of equation (4.9) with a certainty which is fixed but not identically one, we obtain

$$\mathbf{r}(\mathbf{x}) = \mathbf{G}^{-1} \begin{pmatrix} ((\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1)\star\mathbf{f})(\mathbf{x}) \\ \vdots \\ ((\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m)\star\mathbf{f})(\mathbf{x}) \end{pmatrix}, \tag{8.6}$$

where

$$\mathbf{G} = \begin{pmatrix} (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1, \mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_1, \mathbf{b}_m) \\ \vdots & \ddots & \vdots \\ (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m, \mathbf{b}_1) & \dots & (\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_m, \mathbf{b}_m) \end{pmatrix} \tag{8.7}$$

and $\mathbf{c}$ is either $\mathbf{c}_{\text{on}}$ or $\mathbf{c}_{\text{off}}$. Now we would want $\{\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_k\}$ to be Cartesian separable. As before the basis functions are trivially separable and we can limit ourselves to separable applicabilities. The problem comes with the certainty. The x direction is trivially separable from t and y but the latter two are not Cartesian separable. However, they can be decomposed as a sum of two separable products. More precisely we have, reusing the example from (8.4) and (8.5),

$$\begin{aligned} \mathbf{c}_{\text{on}} &= \begin{pmatrix} 1&0&1&0&1\\ 0&1&0&1&0\\ 1&0&1&0&1\\ 0&1&0&1&0\\ 1&0&1&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&1&0&1\\ 0&0&0&0&0\\ 1&0&1&0&1\\ 0&0&0&0&0\\ 1&0&1&0&1 \end{pmatrix} + \begin{pmatrix} 0&0&0&0&0\\ 0&1&0&1&0\\ 0&0&0&0&0\\ 0&1&0&1&0\\ 0&0&0&0&0 \end{pmatrix} \\ &= \begin{pmatrix} 1\\0\\1\\0\\1 \end{pmatrix} \begin{pmatrix} 1&0&1&0&1 \end{pmatrix} + \begin{pmatrix} 0\\1\\0\\1\\0 \end{pmatrix} \begin{pmatrix} 0&1&0&1&0 \end{pmatrix} \end{aligned} \tag{8.8}$$

and similarly

$$\mathbf{c}_{\text{off}} = \begin{pmatrix} 0&1&0&1&0\\ 1&0&1&0&1\\ 0&1&0&1&0\\ 1&0&1&0&1\\ 0&1&0&1&0 \end{pmatrix} = \begin{pmatrix} 1\\0\\1\\0\\1 \end{pmatrix} \begin{pmatrix} 0&1&0&1&0 \end{pmatrix} + \begin{pmatrix} 0\\1\\0\\1\\0 \end{pmatrix} \begin{pmatrix} 1&0&1&0&1 \end{pmatrix}. \tag{8.9}$$

Introducing

$$\mathbf{c}_1 = \begin{pmatrix} 1\\0\\1\\0\\1 \end{pmatrix}, \quad \mathbf{c}_2 = \begin{pmatrix} 0\\1\\0\\1\\0 \end{pmatrix} \tag{8.10}$$

we can rewrite (8.8) and (8.9) more compactly as

$$\mathbf{c}_{\text{on}} = \mathbf{c}_1\mathbf{c}_1^T + \mathbf{c}_2\mathbf{c}_2^T, \tag{8.11}$$

$$\mathbf{c}_{\text{off}} = \mathbf{c}_1\mathbf{c}_2^T + \mathbf{c}_2\mathbf{c}_1^T. \tag{8.12}$$

Notice that the same factors appear in both expressions. This fact can be exploited in the construction of a hierarchical correlator structure similar to the one in figure 4.4. The structure becomes considerably more complex this time, however. Appendix I shows how the resulting structure looks when limited to the ty-plane. A much simpler case occurs when we do polynomial expansion only in the xy-plane, which is further discussed in the next section.

It remains to study the structure of $\mathbf{G}$. As in section 4.3.2 we restrict ourselves to applicabilities which are even along all axes. Both $\mathbf{c}_{\text{on}}$ and $\mathbf{c}_{\text{off}}$ are sufficiently symmetric for elements of $\mathbf{G}$ which were zero in (4.10) to remain zero. Thus we have the same sparsity as before but we get a larger diversity in the non-zero values. It is obvious from the structure that $\mathbf{G}^{-1}$ cannot become less sparse than $\mathbf{G}$. Experiments indicate that it in fact becomes almost, but not quite[5], as sparse as in (4.11). This fact is not critical, however, so we do not prove it.

8.6 One Frame Motion Estimation

Because an interlaced video frame consists of two fields taken from two different time instances, we can estimate motion from a single frame. This is not only a curiousity, but does have some potential real value. Consider a real-time computer vision system where certain kinds of loads forces it to drop occasional frames. This can make it difficult for a displacement estimation module which requires, or at least works best with, two consecutive frames[6]. The advantage of a one-frame algorithm would be that we can be almost certain that it would always get consistent data, i.e. two consecutive fields.

To estimate motion from a single interlaced frame we naturally want to use the displacement estimation algorithm presented in the previous chapter. As input to the algorithm we need the 2D quadratic polynomial expansion in the xy-plane for each field of the frame. In this case the two certainty masks become

$$\mathbf{c}_{\text{on}} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}, \quad \mathbf{c}_{\text{off}} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{8.13}$$

In contrast to (8.4) and (8.5) these are clearly Cartesian separable. As a consequence the products $\{\mathbf{a}\cdot\mathbf{c}\cdot\mathbf{b}_k\}$ used in (8.6) all become separable[7] and can efficiently be computed through 1D correlations by the hierarchical correlator structure in figure 8.5. Since it does not matter what signal values we have at points with certainty zero, we can use the original frame without repacking it. Computing polynomial expansion coefficients corresponding to $\mathbf{c}_{\text{on}}$ at the full vertical resolution gives us every second line for the first field and the complementary lines

[5] One more off-diagonal element appears, which in many cases is very small.

[6] In a well-designed system this would be a non-issue. Real systems tend to have various flaws, however.

[7] Assuming a separable applicability, naturally.

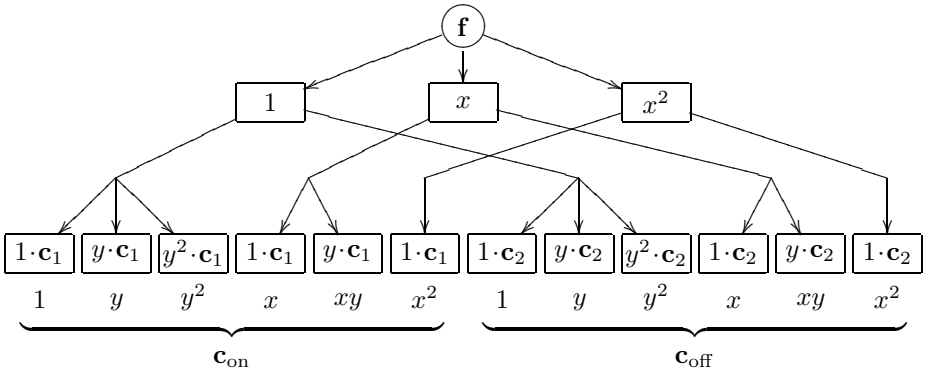


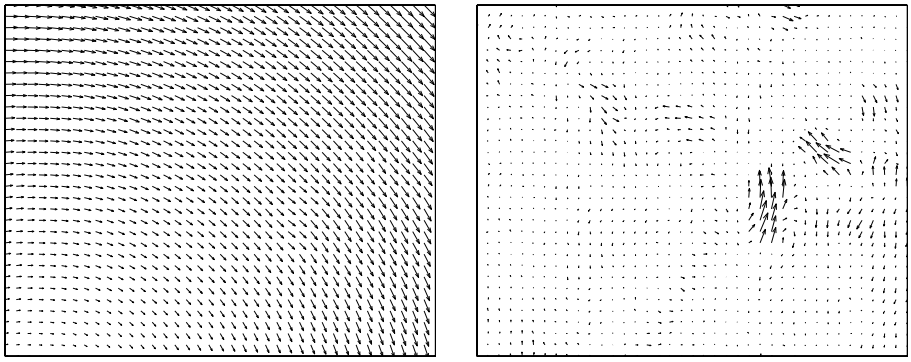
Figure 8.5: Correlator structure for polynomial expansion in the xy-plane for interlaced signals. There is understood to be an applicability factor in each box as well.

for the second field. The expansion coefficients corresponding to $\mathbf{c}_{\text{off}}$ fill in the remaining lines. As a result we obtain the polynomial expansion for both fields at the full vertical resolution.

With this we are ready to use the displacement estimation algorithms from chapter 7. Applying the motion detection algorithm from section 7.10.3 to the frame in figure 8.3 gives the result shown in figure 8.6. This is very similar to figure 7.20 but looks slightly more noisy.

We have also tested the displacement estimation algorithm on an interlaced version of the Yosemite sequence. This was produced by Andersson [3, 4] from the progressive Yosemite sequence through lowpass filtering and vertical downsampling according to the interlaced sampling geometry. We estimated the displacements from the interlaced frame where the bottom field corresponds to the original center frame and the top field to the frame before. Using the same weighting function as in section 7.9.2 and the multi-scale method in two scales[8] we obtain average angular errors of $4.17° \pm 10.45°$ for the affine motion model and $4.72° \pm 9.43°$ for the constant motion model. These can be reduced to $2.38° \pm 4.08°$ and $2.71° \pm 2.76°$ respectively by computing the polynomial expansion with normalized convolution, correctly setting the certainty to zero outside the borders, instead of the fast algorithm described above. The latter results are only marginally worse than the corresponding ones for the original Yosemite sequence, listed in table 7.1. The smallest error reported by Andersson was about 6°.

[8] Interlaced polynomial expansion using Gaussian applicability of sizes 19×19 and 9×9. Standard deviations 2.7 and 1.2 respectively.



(a) (b)

Figure 8.6: Displacement estimated from the single interlaced frame in figure 8.3 and residual displacement field analogous to figure 7.20.

Chapter 9

The Spatial Domain Toolbox

9.1 Introduction

In order to make the theory and algorithms developed in this thesis more accessible to other researchers, Matlab implementations of many of the algorithms have been collected in a package called the Spatial Domain Toolbox. This can be freely downloaded in source form from `http://www.isy.liu.se/~gf/software`.[1]

The package is distributed under the terms of the GNU General Public License. Essentially this allows you to read, use, and modify the code freely. However, if you want to redistribute it, with or without modifications, you must do that under the same license. Contact the author if you wish to negotiate other licensing terms.

9.2 Functions in the Toolbox

Below is a list of the most generally interesting functions in version 2.0 of the toolbox.

- `normconv`
  Normalized convolution with arbitrary number of basis functions and arbitrary dimensionality. Computed according to equation (3.9), using the point by point strategy. Limited to real signals and basis functions.

- `polyexp`
  Polynomial expansion in one to three dimensions. Arbitrary applicability, constant or varying certainty, and arbitrary selection of monomials in the basis. All methods listed in section 4.4 are implemented. Limited to real signals.

- `make_Abc_fast`
  More optimized implementation of polynomial expansion. Limited to Gaus-

[1] If that address should stop working in the future, there are some chances that `http://www.lysator.liu.se/~gunnar` may still provide a link. Otherwise you have to rely on the search engines.

sian applicability, constant certainty, quadratic polynomials, real signals, and one to four dimensions. Implemented through separable correlation.

- `poly_to_tensor`
  Computation of orientation tensors from quadratic polynomial expansion coefficients according to equation (5.19).

- `make_tensors_fast`
  More optimized computation of orientation tensors. Essentially `make_Abc_fast` and `poly_to_tensor` in one box.

- `velocity_from_tensors`
  Estimation of velocity from a tensor field, using the fast algorithm described in section 6.5. Choose between the constant, affine, and eight-parameter motion models. Limited to two spatial dimensions and one temporal.

- `velocity_segment`
  Estimation of velocity from a tensor field, using the segmentation algorithm described in section 6.4. Limited to two spatial dimensions and one temporal.

- `estimate_disparity`
  Estimation of disparity from two images, as described in section 7.3.

- `estimate_displacement`
  Estimation of displacement from two images, as described in chapter 7.

- `one_frame_motion`
  Estimation of displacement from one interlaced image, as described in section 8.6.

Chapter 10

Conclusions and Future Research Directions

10.1 Conclusions

There seems to be a widespread belief that efficient solution of a least squares problem requires a change to an orthogonal basis. It is not uncommon to see a considerable effort spent in constructing such a basis, in particular in discretized multidimensional spaces. The end result is often effective but very inflexible with respect to changes in the geometry and does not easily allow a modified weighting in the least squares problem. One of the main[1] messages of this thesis is that the use of dual vector sets, presented in chapter 2, often allows a solution which is no less efficient but much more flexible, as further demonstrated in chapters 3 and 4.

In general the theory presented in chapter 2 is well worth mastering, with the possible exception of section 2.5, for anyone who encounters least squares problems in finite dimensional spaces. Parts of it are common knowledge of course.

Normalized convolution is a powerful tool for signal processing. Unfortunately it is not yet widely known. It is our hope that the presentation in chapter 3 will help popularize it.

Building on the theory from the two previous chapters, polynomial expansion is developed in chapter 4. This transform is very central to this thesis and is used extensively and with good results in the applications in the following chapters. With efficient computational schemes and the flexibility provided by the applicability and certainty mechanisms, we are of the opinion that polynomial expansion is a strong candidate to become one of the standard transforms in the future.

The applications of polynomial expansion start with estimation of orientation tensors in chapter 5. This is not the only way to compute orientation tensors and the evaluation in section 5.7 is far from sufficient to prove that it is a superior method. It does show excellent results though, in particular when used in the velocity estimation algorithms in chapter 6, and is computationally efficient.

[1] Although possibly somewhat hidden.

Compared to estimation through quadrature filters (see section 5.2.2) an attraction of our method is that the scale can be changed effortlessly without requiring optimization of new filters. An advantage for the quadrature filter method is that it gives a phase-invariant result, which may be important for certain applications. The concept of orientation functionals is theoretically interesting and may help the understanding of how the different orientation tensor estimation methods relate to each other, as well as show the way to new orientation estimation methods. Some preliminary results can be found in [55].

The velocity estimation algorithms in section 6 make direct use of the orientation tensors from the previous chapter. As the evaluation shows they are very successful and form state of the art for velocity estimation on the Yosemite sequence. While that sequence is commonly used and generally considered as a difficult test case for velocity estimation algorithms, one should remember that it does not tell everything. In particular one can assume that the algorithms make good use of the temporal coherence and the relatively large areas with reasonably consistent parametric motion. If your application has motion of that kind our algorithms will probably perform very well. More difficult situations include large motions, requiring very large applicability for the spatiotemporal filters to perceive the motion at all, and very small objects, where there is not enough data for a robust estimation of the motion model parameters. Moreover the fast algorithm can obviously be expected to have some difficulties with motion discontinuities. On the other hand most algorithms have similar weaknesses, so we recommend that you try our algorithms anyway.

The displacement estimation algorithm in chapter 7 is based directly on polynomial expansion. Using only two frames it can handle other types of motion than the velocity estimation algorithms. In particular it can handle motion which is not temporally coherent at all, as in the example in section 7.1, where spatiotemporal filtering is mostly pointless. Using the multi-scale approach it can also handle large motions well. The main disadvantage is that it requires fairly large spatial averaging to produce robust estimates, which is a problem with discontinuities and small objects.

The main point of the analysis of interlaced video signals in chapter 8 is to demonstrate the power of the certainty mechanism. With only a small effort we could adapt our algorithms for use with interlaced signals.

10.2 Future Research Directions

Many possibilities for extensions of the theory and improvements of the algorithms have been discussed throughout the thesis. In addition to those there are some more ideas we want to mention.

The theory for polynomial expansion seems reasonably mature but there are some open questions regarding multi-scale polynomial expansion, especially for non-constant certainty. There is probably more to be found out about approximative methods as well. To be consistent with the signal/certainty principle polynomial expansion should produce an output certainty. Although this can be

derived from normalized convolution, there are probably optimizations available for a polynomial basis.

The concept of phase functionals discussed in section 5.9 seems promising but is so far lacking a good application where it can be tested.

The velocity estimation algorithms may have some potential for improvements by combining the parameter estimation method described in section 6.3.3 with some other techniques to handle discontinuities. Also the segmentation algorithm might be improved by trying to detect object edges from more cues than motion.

The displacement estimation algorithm definitely has potential for improvement by combining it with some technique to handle discontinuities. One obvious possibility is to adapt the segmentation algorithm described in section 6.4 for use with the displacement algorithm. This is fairly straighforward but requires some implementation work.

More generally it would of course be interesting to find more areas where polynomial expansion can be applied. Appendix J describes a possible approach to adaptive filtering.

Finally one should not forget the initial motivation for doing all algorithm development in the spatial domain, namely being able to adapt the algorithms to irregularly sampled signals, see section 1.1. This research direction would still be interesting to pursue.

Appendices

A A Matrix Inversion Lemma

To prove that the matrix $\mathbf{G}$ in equation (4.10), section 4.3.2, has an inverse given by equation (4.11), we need the following lemma:

Lemma A.1. *Provided that $a \neq 0$, $c \neq d$, and $ad = b^2$, the $(n+1) \times (n+1)$ matrix*

$$\mathbf{M} = \begin{pmatrix} a & b & b & \dots & b \\ b & c & d & \dots & d \\ b & d & c & \dots & d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b & d & d & \dots & c \end{pmatrix} \tag{A.1}$$

has an inverse of the form

$$\mathbf{M}^{-1} = \begin{pmatrix} \mathfrak{a} & \mathfrak{e} & \mathfrak{e} & \dots & \mathfrak{e} \\ \mathfrak{e} & \mathfrak{c} & 0 & \dots & 0 \\ \mathfrak{e} & 0 & \mathfrak{c} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathfrak{e} & 0 & 0 & \dots & \mathfrak{c} \end{pmatrix}. \tag{A.2}$$

Proof. Inserting (A.1) and (A.2) into the definition of the inverse, $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$, we get the five distinct equations

$$a\mathfrak{a} + nb\mathfrak{e} = 1, \tag{A.3}$$
$$a\mathfrak{e} + b\mathfrak{c} = 0, \tag{A.4}$$
$$b\mathfrak{a} + c\mathfrak{e} + (n-1)d\mathfrak{e} = 0, \tag{A.5}$$
$$b\mathfrak{e} + c\mathfrak{c} = 1, \tag{A.6}$$
$$b\mathfrak{e} + d\mathfrak{c} = 0, \tag{A.7}$$

which can easily be verified to have the explicit solution

$$\begin{aligned}\mathfrak{a} &= \frac{1}{a}\left(1+\frac{nd}{c-d}\right),\\ \mathfrak{c} &= \frac{1}{c-d},\\ \mathfrak{e} &= -\frac{b}{a(c-d)} = -\frac{d}{b(c-d)}.\end{aligned} \tag{A.8}$$

□

Since $\mathbf{G}$ can be partitioned into one diagonal block and one block with the structure given by $\mathbf{M}$ in lemma A.1, the stated structure of $\mathbf{G}^{-1}$ follows immediately if we can show that the conditions of the lemma are satisfied. The components of $\mathbf{M}$ are given by

$$a = (\mathbf{a}\cdot\mathbf{b}_1, \mathbf{b}_1), \tag{A.9}$$

$$b = (\mathbf{a}\cdot\mathbf{b}_1, \mathbf{b}_{x_i^2}), \tag{A.10}$$

$$c = (\mathbf{a}\cdot\mathbf{b}_{x_i^2}, \mathbf{b}_{x_i^2}), \tag{A.11}$$

$$d = (\mathbf{a}\cdot\mathbf{b}_{x_i^2}, \mathbf{b}_{x_j^2}), \tag{A.12}$$

so it is clear that $a > 0$. That $c \neq d$ follows by necessity from the assumption that the basis functions are linearly independent, section 4.2. The final requirement that $ad = b^2$ relies on the condition set for the applicability[1]

$$a(\mathbf{x}) = a_1(x_1)a_1(x_2)\ldots a_1(x_N). \tag{A.13}$$

Now we have

$$a = \sum_{x_1,\ldots,x_N} a(\mathbf{x}) = \prod_{k=1}^{N}\left(\sum_{x_k} a_1(x_k)\right) = \left(\sum_{x_1} a_1(x_1)\right)^N, \tag{A.14}$$

$$\begin{aligned} b &= \sum_{x_1,\ldots,x_N} a(\mathbf{x})x_i^2 = \left(\sum_{x_i} a_1(x_i)x_i^2\right)\prod_{k\neq i}\left(\sum a_1(x_k)\right)\\ &= \left(\sum_{x_1} a_1(x_1)x_1^2\right)\left(\sum_{x_1} a_1(x_1)\right)^{N-1}, \end{aligned} \tag{A.15}$$

$$\begin{aligned} d &= \sum_{x_1,\ldots,x_N} a(\mathbf{x})x_i^2x_j^2 = \left(\sum_{x_i} a_1(x_i)x_i^2\right)\left(\sum_{x_j} a_1(x_j)x_j^2\right)\prod_{k\neq i,j}\left(\sum a_1(x_k)\right)\\ &= \left(\sum_{x_1} a_1(x_1)x_1^2\right)^2\left(\sum_{x_1} a_1(x_1)\right)^{N-2} \end{aligned} \tag{A.16}$$

and it is clear that $ad = b^2$.

[1] We apologize that the symbol a happens to be used in double contexts here and trust that the reader manages to keep them apart.

B Cartesian Separable and Isotropic Functions

As we saw in section 4.3.2, a desirable property of the applicability is to simultaneously be Cartesian separable and isotropic. In this appendix we show that the only interesting class of functions having this property is the Gaussians.

Lemma B.1. *Assume that* $f : \mathcal{R}^N \longrightarrow \mathcal{R}$, $N \geq 2$, *is Cartesian separable,*

$$f(\mathbf{x}) = f_1(x_1)f_2(x_2)\dots f_N(x_N), \quad \textit{some } \{f_k : \mathcal{R} \longrightarrow \mathcal{R}\}_{k=1}^N, \tag{B.1}$$

isotropic,

$$f(\mathbf{x}) = g(\mathbf{x}^T\mathbf{x}), \quad \textit{some } g : \mathcal{R}_+ \cup \{0\} \longrightarrow \mathcal{R}, \tag{B.2}$$

and partially differentiable. Then f *must be of the form*

$$f(\mathbf{x}) = Ae^{C\mathbf{x}^T\mathbf{x}}, \tag{B.3}$$

for some real constants A *and* C.

Proof. We first assume that f is zero for some $\mathbf{x}$. Then at least one factor in (B.1) is zero and by varying the remaining coordinates it follows that

$$g(t) = 0, \quad \text{all } t \geq \alpha^2, \tag{B.4}$$

where α is the value of the coordinate in the zero factor. By taking

$$\mathbf{x} = \frac{\alpha}{\sqrt{N}}\begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix}^T \tag{B.5}$$

we can repeat the argument to get

$$g(t) = 0, \quad \text{all } t \geq \frac{\alpha^2}{N}. \tag{B.6}$$

and continuing like this we find that $g(t) = 0$, all $t > 0$, and since f is partially differentiable there cannot be a point discontinuity at the origin, so f must be identically zero. This is clearly a valid solution.

If instead f is nowhere zero we can compute the partial derivatives as

$$\frac{\frac{\partial}{\partial x_k}f(\mathbf{x})}{f(\mathbf{x})} = \frac{f_k'(x_k)}{f_k(x_k)} = 2x_k\frac{g'(\mathbf{x}^T\mathbf{x})}{g(\mathbf{x}^T\mathbf{x})}, \quad k = 1, 2, \dots, N. \tag{B.7}$$

Restricting ourselves to one of the hyper quadrants, so that all $x_k \neq 0$, we get

$$\frac{g'(\mathbf{x}^T\mathbf{x})}{g(\mathbf{x}^T\mathbf{x})} = \frac{f_1'(x_1)}{2x_1f_1(x_1)} = \frac{f_2'(x_2)}{2x_2f_2(x_2)} = \dots = \frac{f_N'(x_N)}{2x_Nf_N(x_N)}, \tag{B.8}$$

which is possible only if they all have a common constant value C. Hence we get g from the differential equation

$$g'(t) = Cg(t) \tag{B.9}$$

with the solution

$$g(t) = Ae^{Ct}. \tag{B.10}$$

It follows that f in each hyper quadrant must have the form

$$f(\mathbf{x}) = Ae^{C\mathbf{x}^T\mathbf{x}} \tag{B.11}$$

and in order to get isotropy, the constants must be the same for all hyper quadrants. The case $A = 0$ corresponds to the identically zero solution. □

A weakness of this result is the condition that f be partially differentiable, which is not a natural requirement of an applicability function. If we remove this condition it is easy to find one new solution, which is zero everywhere except at the origin. What is not so easy to see however, and quite contra-intuitive, is that there also exist solutions which are discontinuous and everywhere positive. To construct these we need another lemma.

Lemma B.2. *There do exist* discontinuous *functions* $L : \mathcal{R} \longrightarrow \mathcal{R}$ *which are additive, i.e.*

$$L(x + y) = L(x) + L(y), \quad x, y \in \mathcal{R}. \tag{B.12}$$

Proof. See [83] or [32]. □

With L from the lemma we get a Cartesian separable and isotropic function by the construction

$$f(\mathbf{x}) = e^{L(\mathbf{x}^T\mathbf{x})}. \tag{B.13}$$

This function is very bizarre, however, because it has to be discontinuous at every point and unbounded in every neighborhood of every point. It is also completely useless as an applicability since it is unmeasurable, i.e. it cannot be integrated.

To eliminate this kind of strange solutions it is sufficient to introduce some very weak regularity constraints[2] on f. Unfortunately the proofs become very technical if we want to have a bare minimum of regularity. Instead we explore what can be accomplished with a regularization approach.

Let the functions ϕ_σ and Φ_σ be normalized Gaussians with standard deviation σ in one and N dimensions respectively,

$$\phi_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \tag{B.14}$$

$$\Phi_\sigma(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{N}{2}}\sigma^N} e^{-\frac{\mathbf{x}^T\mathbf{x}}{2\sigma^2}} = \prod_{k=1}^{N} \phi_\sigma(x_k). \tag{B.15}$$

We now make the reasonable assumption that f is regular enough to be convolved with Gaussians,[3]

$$f_\sigma(\mathbf{x}) = (f * \Phi_\sigma)(\mathbf{x}) = \int_{\mathcal{R}^N} f(\mathbf{x} - \mathbf{y})\Phi_\sigma(\mathbf{y})\, d\mathbf{y}, \quad \sigma > 0. \tag{B.16}$$

[2] See e.g. [32] and exercise 9.18 in [75].

[3] This mainly requires f to be locally integrable. The point is, however, that it would be useless as an applicability if it could not be convolved with Gaussians.

The convolved functions retain the properties of f to be Cartesian separable and isotropic. The first property can be verified by

$$\begin{aligned} f_\sigma(\mathbf{x}) &= \int_{\mathcal{R}^N} f_1(x_1 - y_1) \dots f_N(x_N - y_N)\, \phi_\sigma(y_1) \dots \phi_\sigma(y_N)\, d\mathbf{y} \\ &= \prod_{k=1}^{N} \int_{\mathcal{R}} f_k(x_k - y_k) \phi_\sigma(y_k)\, dy_k. \end{aligned} \tag{B.17}$$

To show the second property we notice that isotropy is equivalent to rotation invariance, i.e. for an arbitrary rotation matrix $\mathbf{R}$ we have $f(\mathbf{Rx}) = f(\mathbf{x})$. Since the Gaussians are rotation invariant too, we have

$$\begin{aligned} f_\sigma(\mathbf{Rx}) &= \int_{\mathcal{R}^N} f(\mathbf{Rx} - \mathbf{y}) \Phi_\sigma(\mathbf{y})\, d\mathbf{y} \\ &= \int_{\mathcal{R}^N} f(\mathbf{Rx} - \mathbf{Ru}) \Phi_\sigma(\mathbf{Ru})\, d\mathbf{u} \\ &= \int_{\mathcal{R}^N} f(\mathbf{x} - \mathbf{u}) \Phi_\sigma(\mathbf{u})\, d\mathbf{u} = f_\sigma(\mathbf{x}). \end{aligned} \tag{B.18}$$

Another property that the convolved functions obtain is a high degree of regularity. Without making additional assumptions on the regularity of f, f_σ is guaranteed to be infinitely differentiable because Φ_σ has that property. This means that lemma B.1 applies to f_σ, which therefore has to be a Gaussian.

To connect the convolved functions to f itself we notice that the Gaussians Φ_σ approach the Dirac distribution as σ approaches zero; they become more and more concentrated to the origin. As a consequence the convolved functions f_σ approach f and in the limit we find that f has to be a Gaussian, at least almost everywhere.[4]

Another way to reach this conclusion is to assume that f can be Fourier transformed. Then equation (B.16) turns into $\hat{f}_\sigma(\mathbf{u}) = \hat{f}(\mathbf{u})\hat{\Phi}_\sigma(\mathbf{u})$, so that $\hat{f}(\mathbf{u}) = \frac{\hat{f}_\sigma(\mathbf{u})}{\hat{\Phi}_\sigma(\mathbf{u})}$ is a quotient of two Gaussians and hence itself a Gaussian. Inverse Fourier transformation gives the desired result.

[4] Almost everywhere is quite sufficient because the applicability is only used in integrals.

C Correlator Structure for Separable Normalized Convolution

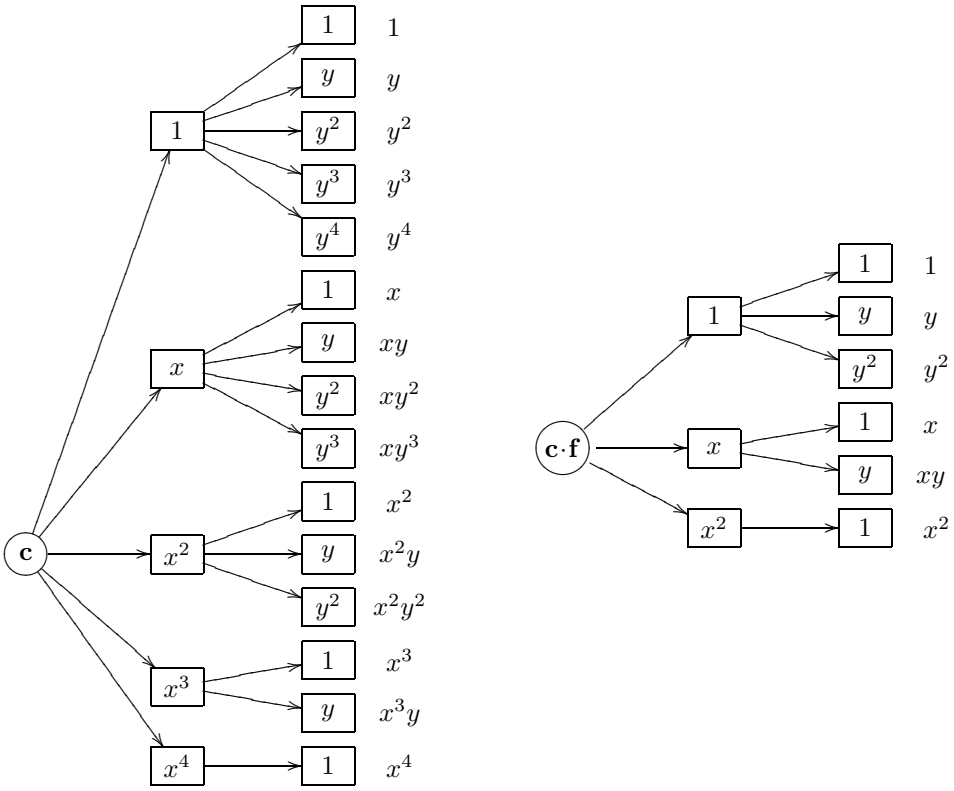


Figure C.1: Correlator structure for computation of quadratic polynomial expansion in 2D, using the separable normalized convolution method described in section 4.4. There is understood to be an applicability factor in each box as well.

D Gaussian Correlation Formulas

This appendix adds details for some of the calculations in section 4.5. As in that section we let

$$g_1(x) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{x^2}{2\sigma_1^2}}, \tag{D.1}$$

$$g_2(x) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{x^2}{2\sigma_2^2}}, \tag{D.2}$$

$$g_3(x) = \frac{1}{\sqrt{2\pi}\sigma_3} e^{-\frac{x^2}{2\sigma_3^2}}, \tag{D.3}$$

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2. \tag{D.4}$$

As a preparation we state the well known integrals

$$\int e^{-at^2}\, dt = \sqrt{\frac{\pi}{a}}, \tag{D.5}$$

$$\int te^{-at^2}\, dt = 0, \tag{D.6}$$

$$\int t^2 e^{-at^2}\, dt = \frac{1}{2a}\sqrt{\frac{\pi}{a}}, \tag{D.7}$$

$$\int t^3 e^{-at^2}\, dt = 0, \tag{D.8}$$

$$\int t^4 e^{-at^2}\, dt = \frac{3}{4a^2}\sqrt{\frac{\pi}{a}}, \tag{D.9}$$

where $a > 0$ and all integration, as in the rest of this appendix, is from $-\infty$ to ∞. Next we study integrals of the form

$$I_k(s) = \int (s-u)^k e^{-a(s-u)^2} e^{-bu^2}\, du, \quad a, b > 0. \tag{D.10}$$

To simplify this we make the change of variables $u = t + \frac{as}{a+b}$, giving

$$\begin{aligned} I_k(s) &= \int (s - \frac{a}{a+b}s - t)^k e^{-a(s-\frac{a}{a+b}s-t)^2 - b(t+\frac{a}{a+b}s)^2}\, dt \\ &= \int (\frac{b}{a+b}s - t)^k e^{-(a+b)t^2 - \frac{ab}{a+b}s^2}\, dt \\ &= e^{-\frac{ab}{a+b}s^2} \int (\frac{b}{a+b}s - t)^k e^{-(a+b)t^2}\, dt. \end{aligned} \tag{D.11}$$

For $k = 0, 1, 2$ we obtain, with the help of (D.5)–(D.7),

$$I_0(s) = \sqrt{\frac{\pi}{a+b}} e^{-\frac{ab}{a+b}s^2}, \tag{D.12}$$

$$I_1(s) = \frac{bs}{a+b}\sqrt{\frac{\pi}{a+b}} e^{-\frac{ab}{a+b}s^2}, \tag{D.13}$$

$$I_2(s) = \left(\frac{b^2s^2}{(a+b)^2} + \frac{1}{2(a+b)}\right)\sqrt{\frac{\pi}{a+b}} e^{-\frac{ab}{a+b}s^2}. \tag{D.14}$$

Now we are ready to take on the expressions in equation (4.17),

$$\begin{aligned} J_k(x) &= (x^k g_2 \star (g1 \star f))(x) \\ &= \int t^k g_2(t) \int g_1(u) f(u+t+x)\, du\, dt \\ &= \int\int t^k g_2(t) g_1(u) f(u+t+x)\, du\, dt \\ &= \int\int (s-u)^k g_2(s-u) g_1(u) f(s+x)\, du\, ds \\ &= \frac{1}{2\pi\sigma_1\sigma_2}\int\int (s-u)^k e^{-\frac{(s-u)^2}{2\sigma_2^2}} e^{-\frac{u^2}{2\sigma_1^2}}\, du f(s+x)\, ds. \end{aligned} \tag{D.15}$$

Using equations (D.12)–(D.14) we obtain

$$\begin{aligned} J_0(x) &= \frac{1}{2\pi\sigma_1\sigma_2}\int \sqrt{\frac{2\pi\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} e^{-\frac{s^2}{2(\sigma_1^2+\sigma_2^2)}} f(s+x)\, ds \\ &= \int g_3(s) f(s+x)\, ds = (g_3 \star f)(x), \end{aligned} \tag{D.16}$$

$$\begin{aligned} J_1(x) &= \frac{1}{2\pi\sigma_1\sigma_2}\int \frac{s\sigma_2^2}{\sigma_1^2+\sigma_2^2}\sqrt{\frac{2\pi\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} e^{-\frac{s^2}{2(\sigma_1^2+\sigma_2^2)}} f(s+x)\, ds \\ &= \int \left(\frac{\sigma_2}{\sigma_3}\right)^2 s g_3(s) f(s+x)\, ds = \left(\frac{\sigma_2}{\sigma_3}\right)^2 (x g_3 \star f)(x), \end{aligned} \tag{D.17}$$

$$\begin{aligned} J_2(x) &= \frac{1}{2\pi\sigma_1\sigma_2}\int \left(\frac{s^2\sigma_2^4}{(\sigma_1^2+\sigma_2^2)^2} + \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)\sqrt{\frac{2\pi\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} e^{-\frac{s^2}{2(\sigma_1^2+\sigma_2^2)}} f(s+x)\, ds \\ &= \int \left(\left(\frac{\sigma_2}{\sigma_3}\right)^4 s^2 + \frac{\sigma_1^2\sigma_2^2}{\sigma_3^2}\right) g_3(s) f(s+x)\, ds \\ &= \left(\frac{\sigma_2}{\sigma_3}\right)^4 (x^2 g_3 \star f)(x) + \left(\frac{\sigma_1\sigma_2}{\sigma_3}\right)^2 (g_3 \star f)(x). \end{aligned} \tag{D.18}$$

In conclusion we have

$$g_2 \star (g_1 \star f) = g_3 \star f, \tag{D.19}$$

$$xg_2 \star (g_1 \star f) = \left(\frac{\sigma_2}{\sigma_3}\right)^2 (xg_3 \star f), \tag{D.20}$$

$$x^2 g_2 \star (g_1 \star f) = \left(\frac{\sigma_2}{\sigma_3}\right)^4 (x^2 g_3 \star f) + \sigma_1^2 \left(\frac{\sigma_2}{\sigma_3}\right)^2 (g_3 \star f). \tag{D.21}$$

Furthermore (D.5)–(D.9) directly yield the inner products

$$(g_2, 1) = 1, \tag{D.22}$$

$$(g_2, x) = 0, \tag{D.23}$$

$$(g_2, x^2) = \sigma_2^2, \tag{D.24}$$

$$(g_2, x^3) = 0, \tag{D.25}$$

$$(g_2, x^4) = 3\sigma_2^4. \tag{D.26}$$

E Binomial Identities

This appendix proves the binomial identities (4.24)–(4.26) in section 4.6.2. We can restate these directly in terms of binomial coefficients as

$$\binom{2n}{n+k} = \binom{2n-2}{n+k-2} + 2\binom{2n-2}{n+k-1} + \binom{2n-2}{n+k}, \tag{E.1}$$

$$k\binom{2n}{n+k} = n\binom{2n-2}{n+k-2} - n\binom{2n-2}{n+k}, \tag{E.2}$$

$$k^2\binom{2n}{n+k} = n^2\binom{2n-2}{n+k-2} - 2n(n-1)\binom{2n-2}{n+k-1} + n^2\binom{2n-2}{n+k}, \tag{E.3}$$

which we want to show for $n > 0$.

Using standard binomial identities from [36], we transform the right hand side binomial coefficients,

$$\binom{2n-2}{n+k-2} = \frac{n+k-1}{2n-1}\binom{2n-1}{n+k-1} = \frac{(n+k-1)(n+k)}{2n(2n-1)}\binom{2n}{n+k}, \tag{E.4}$$

$$\begin{aligned}\binom{2n-2}{n+k-1} &= \frac{n+k}{2n-1}\binom{2n-1}{n+k} = \frac{n+k}{2n-1}\binom{2n-1}{n-k-1} \\ &= \frac{(n+k)(n-k)}{2n(2n-1)}\binom{2n}{n-k} = \frac{(n+k)(n-k)}{2n(2n-1)}\binom{2n}{n+k},\end{aligned} \tag{E.5}$$

$$\begin{aligned}\binom{2n-2}{n+k} &= \binom{2n-2}{n-k-2} = \frac{n-k-1}{2n-1}\binom{2n-1}{n-k-1} \\ &= \frac{(n-k-1)(n-k)}{2n(2n-1)}\binom{2n}{n-k} = \frac{(n-k-1)(n-k)}{2n(2n-1)}\binom{2n}{n+k}.\end{aligned} \tag{E.6}$$

After inserting (E.4)–(E.6) into (E.1)–(E.3) and eliminating the factor $\binom{2n}{n+k}$, it suffices to show that

$$1 = \frac{(n+k-1)(n+k) + 2(n+k)(n-k) + (n-k-1)(n-k)}{2n(2n-1)}, \tag{E.7}$$

$$k = \frac{n(n+k-1)(n+k) - n(n-k-1)(n-k)}{2n(2n-1)}, \tag{E.8}$$

$$k^2 = \frac{n^2(n+k-1)(n+k) - 2n(n-1)(n+k)(n-k) + n^2(n-k-1)(n-k)}{2n(2n-1)}, \tag{E.9}$$

which is easily verified[5].

[5] Please excuse the ugliness of this proof. It can be done in a more constructive way which naturally extends to higher exponents k^p, but then it does not fit on this page.

F Angular RMS Error

In this appendix we verify the equivalence between the expressions (5.31) and (5.32) for the angular RMS error in section 5.7.

Starting with

$$\Delta\phi = \arcsin\left(\sqrt{\frac{1}{2L}\sum_{l=1}^{L}\|\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T\|^2}\right), \tag{F.1}$$

we expand the squared Frobenius norm, using the relation $\|\mathbf{T}\|^2 = \mathrm{tr}\,(\mathbf{T}^T\mathbf{T})$, to get

$$\begin{aligned}\|\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T\|^2 &= \mathrm{tr}\big((\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T)^T(\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T)\big)\\ &= \mathrm{tr}\big(\hat{\mathbf{x}}\hat{\mathbf{x}}^T - (\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)\hat{\mathbf{x}}\hat{\mathbf{e}}_1^T - (\hat{\mathbf{e}}_1^T\hat{\mathbf{x}})\hat{\mathbf{e}}_1\hat{\mathbf{x}}^T + \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T\big).\end{aligned} \tag{F.2}$$

To simplify this expression we use the fact that the trace operator is linear and that $\mathrm{tr}(\mathbf{a}\mathbf{b}^T) = \mathrm{tr}(\mathbf{b}^T\mathbf{a})$. Thus we have

$$\|\hat{\mathbf{x}}\hat{\mathbf{x}}^T - \hat{\mathbf{e}}_1\hat{\mathbf{e}}_1^T\|^2 = 1 - (\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2 - (\hat{\mathbf{e}}_1^T\hat{\mathbf{x}})^2 + 1 = 2(1 - (\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2) \tag{F.3}$$

and continuing with the original expression,

$$\begin{aligned}\Delta\phi &= \arcsin\left(\sqrt{\frac{1}{2L}\sum_{l=1}^{L}2(1 - (\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2)}\right)\\ &= \arcsin\left(\sqrt{1 - \frac{1}{L}\sum_{l=1}^{L}(\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2}\right)\\ &= \arccos\left(\sqrt{\frac{1}{L}\sum_{l=1}^{L}(\hat{\mathbf{x}}^T\hat{\mathbf{e}}_1)^2}\right).\end{aligned} \tag{F.4}$$

G Removing the Isotropic Part of a 3D Tensor

To remove the isotropic part of a 3D tensor we need to compute the smallest eigenvalue of a symmetric and positive semidefinite 3×3 matrix T. There are a number of ways to do this, including inverse iterations and standard methods for eigenvalue factorization. Given the small size of the matrix in this case, however, we additionally have the option of computing the eigenvalues algebraically, since these can be expressed as the roots of the third degree characteristic polynomial, $p(\lambda) = \det(\mathbf{T} - \lambda\mathbf{I})$.

To find the solutions to $z^3 + az^2 + bz + c = 0$ we first remove the quadratic term with the translation $z = x - \frac{a}{3}$, yielding

$$x^3 + px + q = 0. \tag{G.1}$$

It is well known that the solutions to this equation can be given by Cardano's formula [88],

$$D = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^2, \tag{G.2}$$

$$u = \sqrt[3]{-\frac{q}{2} + \sqrt{D}}, \tag{G.3}$$

$$v = \sqrt[3]{-\frac{q}{2} - \sqrt{D}}, \tag{G.4}$$

$$x_1 = u + v, \tag{G.5}$$

$$x_2 = -\frac{u+v}{2} + \frac{u-v}{2} i\sqrt{3}, \tag{G.6}$$

$$x_3 = -\frac{u+v}{2} - \frac{u-v}{2} i\sqrt{3}. \tag{G.7}$$

Unfortunately this formula leads to some complications in the choice of the complex cubic roots if $D < 0$, which happens exactly when we have three distinct real roots. Since we have a symmetric and positive semidefinite matrix we know a priori that all eigenvalues are real and non-negative.[6]

A better approach in this case, still following [88], is to make the scaling $x = \sqrt{-\frac{4p}{3}}\, y$, leading to

$$4y^3 - 3y = \frac{3q}{p\sqrt{-\frac{4p}{3}}}. \tag{G.8}$$

Taking advantage of the identity $\cos 3\alpha = 4\cos^3\alpha - 3\cos\alpha$, we make the substitution $y = \cos\alpha$ to get the equation

$$\cos 3\alpha = \frac{3q}{p\sqrt{-\frac{4p}{3}}}, \tag{G.9}$$

[6] For the following discussion it is sufficient that we have a symmetric matrix and thus real eigenvalues.

where the right hand side is guaranteed to have an absolute value less than or equal to one if all the roots are indeed real. Hence it is clear that we obtain the three real solutions to (G.1) from

$$\beta = \sqrt{-\frac{4p}{3}}, \tag{G.10}$$

$$\alpha = \frac{1}{3}\arccos\frac{3q}{p\beta}, \tag{G.11}$$

$$x_1 = \beta\cos\alpha, \tag{G.12}$$

$$x_2 = \beta\cos\left(\alpha - \frac{2\pi}{3}\right), \tag{G.13}$$

$$x_3 = \beta\cos\left(\alpha + \frac{2\pi}{3}\right). \tag{G.14}$$

Furthermore, since we have $0 \leq \alpha \leq \frac{\pi}{3}$, it follows that $x_1 \geq x_2 \geq x_3$.

In terms of the tensor $\mathbf{T}$, the above discussion leads to the following algorithm for removal of the isotropic part:

1. Remove the trace of $\mathbf{T}$ by computing

$$\mathbf{T}' = \mathbf{T} - \frac{\operatorname{tr}\mathbf{T}}{3}\mathbf{I} = \begin{pmatrix} a & d & e \\ d & b & f \\ e & f & c \end{pmatrix}. \tag{G.15}$$

   This is equivalent to removing the quadratic term from the characteristic polynomial.

2. The eigenvalues of $\mathbf{T}'$ are now given as the solutions to $x^3 + px + q = 0$, where

$$p = ab + ac + bc - d^2 - e^2 - f^2, \tag{G.16}$$

$$q = af^2 + be^2 + cd^2 - 2def - abc. \tag{G.17}$$

3. Let

$$\beta = \sqrt{-\frac{4p}{3}}, \tag{G.18}$$

$$\alpha = \frac{1}{3}\arccos\frac{3q}{p\beta}, \tag{G.19}$$

   so that the eigenvalues of $\mathbf{T}'$ are given by (G.12)–(G.14).

4. Let

$$x_3 = \beta\cos\left(\alpha + \frac{2\pi}{3}\right) \tag{G.20}$$

   and compute the isotropy compensated tensor $\mathbf{T}''$ as

$$\mathbf{T}'' = \mathbf{T}' - x_3\mathbf{I}. \tag{G.21}$$

It should be noted that this method may numerically be somewhat less accurate than standard methods for eigenvalue factorization. For the current application, however, this is not an issue at all.[7]

A slightly different formula for the eigenvalues of a real, symmetric 3×3 matrix can be found in [79] and a closed form formula for eigenvectors as well as eigenvalues in [14].

[7] Except making sure that the magnitude of the argument to the arccosine is not very slightly larger than one.

H Elementary Stereo Geometry

We consider the elementary stereo geometry obtained by setting up two identical pinhole cameras in the canonical configuration shown in figure H.1[8]. This configuration is characterized by the baseline, the line connecting the two optical centers, being aligned with the x-axes of the cameras and both optical axes being parallel. The world coordinate system is placed with its origin between the two cameras, so that we have the origin of the left camera at world coordinates $(-h, 0, f)$ and the origin of the right camera at $(h, 0, f)$, where $2h$ is the distance between the optical centers of the two cameras and f is the focal distance. Using similar triangles we obtain the two relations[9]

$$\frac{h+X}{Z} = \frac{x_l}{f}, \tag{H.1}$$

$$\frac{h-X}{Z} = \frac{-x_r}{f}. \tag{H.2}$$

Summing these gives us

$$\frac{2h}{Z} = \frac{x_l - x_r}{f}. \tag{H.3}$$

The quantity $d = x_l - x_r$ is commonly called disparity so we have the following two relations between depth and disparity,

$$Z = \frac{2hf}{d}, \tag{H.4}$$

$$d = \frac{2hf}{Z}. \tag{H.5}$$

We also notice that from (H.5) it follows that if we know a priori the minimum and maximum distances to objects in the scene, we can also a priori compute upper and lower bounds for the disparity,

$$d_{\min} = \frac{2hf}{Z_{\max}}, \tag{H.6}$$

$$d_{\max} = \frac{2hf}{Z_{\min}}. \tag{H.7}$$

Even if we do not know $Z_{\max}$, we still have the relation $d \geq 0$.

[8] Having the image planes in front of the optical centers is physically incorrect but more convenient to work with.

[9] Notice that the geometry guarantees that the y coordinates in the left and right images are identical.

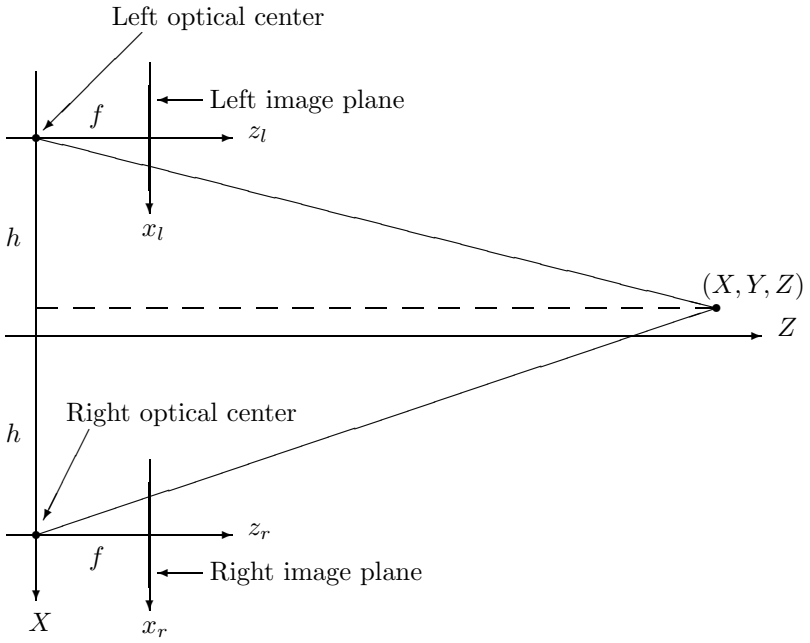


Figure H.1: Canonical stereo configuration.

I Correlator Structure for Interlaced Signals

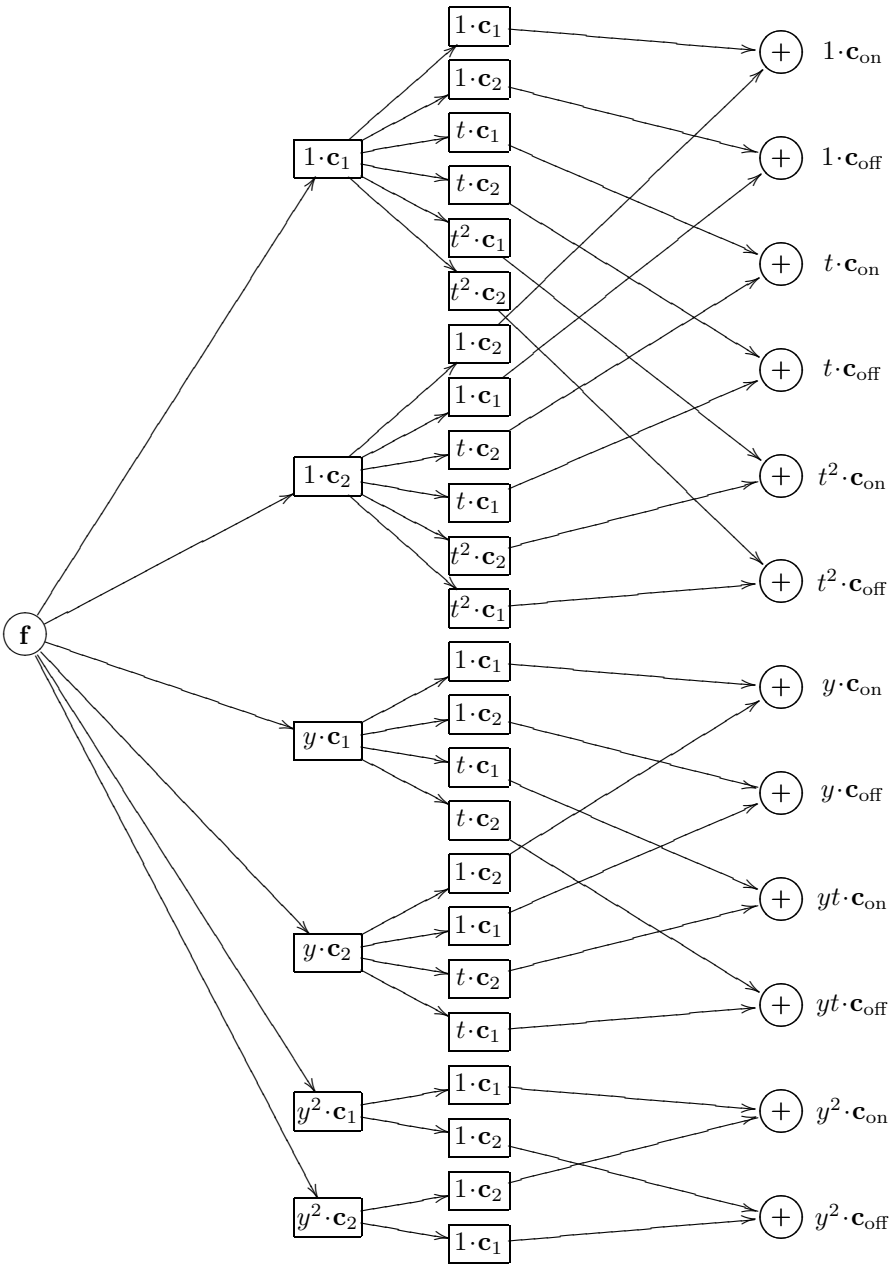


Figure I.1: Correlator structure for computation of polynomial expansion in the ty-plane for interlaced video signals, described in section 8.5. There is understood to be an applicability factor in each box as well.

J An Adaptive Filtering Approach

The idea of adaptive filtering, as described in [40], is to apply a space-variant filter to a signal, where the filter shape at each point is determined by the local orientation. In order for this operation to be practically feasible, it is required that the different filter shapes can be constructed as linear combinations of a small set of space-invariant filters. Without going into the depth of the presentation in [40], we show in this section how quadratic polynomial expansion can be used for adaptive highpass filtering with applications to directed noise and image degradation.

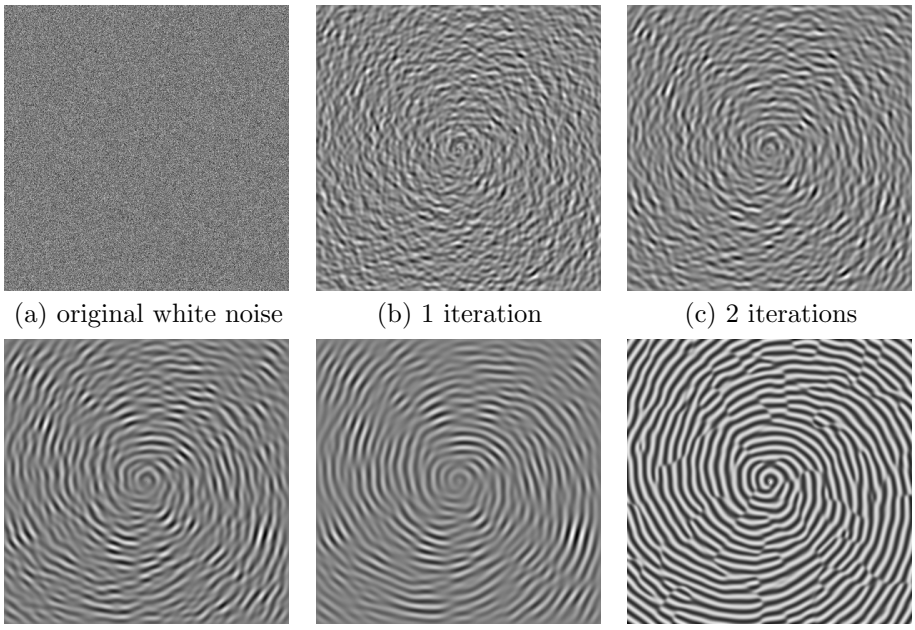
The key observation can be made from figure 4.3, where we can see that the dual basis functions used to compute the expansion coefficients for the x^2 and y^2 basis functions in fact are directed highpass filters along the two axes.[10] Together with the dual basis function corresponding to xy we can construct rotations of this highpass filter to any direction $(\alpha\ \beta)^T$, $\alpha^2 + \beta^2 = 1$, by noting that $(\alpha x + \beta y)^2 = \alpha^2 x^2 + 2\alpha\beta xy + \beta^2 y^2$. Hence the response of a directed highpass filter in any direction can be computed as a linear combination of the expansion coefficients for the second degree basis functions.

Directed noise is obtained by applying adaptive filtering to white noise. The result is noise with a tendency to be oriented in the same way as the orientation field used to control the adaptive filtering. To increase the effect we can iterate this procedure a number of times. In figure J.1 we adapt white noise to an orientation field defined so that the orientation at each point is the radius vector rotated by 10 degrees, with the geometric interpretation of a spiral pattern. We can see that this pattern becomes more distinct with each iteration, although there is also a significant element of very low frequency noise. The latter can be almost completely eliminated by a local amplitude equalization, giving the final result in figure J.1(f). The scale of the pattern is directly controlled by the standard deviation of the Gaussian applicability.

An amusing application of directed noise is to control the filters with the estimated orientation field of an actual image; a severe form of image degradation. In figure J.2 the orientation field of the well-known Lena image has been computed, lowpass filtered and then used to control the directed noise process, with 20 iterations and amplitude equalization.[11] Figure J.3 shows the result of a more intricate variation of the same theme, involving multiple scales and separate processing of each color component. The full color image can be found on the cover of [23].

[10] Notice, however, that we had better negate these filters to avoid an unnecessary 180° phase shift.

[11] Only the largest eigenvector of each tensor has been used to determine the orientation. Cf. figure 3.3 for somewhat less degraded versions of the image.



(a) original white noise (b) 1 iteration (c) 2 iterations

(d) 5 iterations (e) 10 iterations (f) equalized

Figure J.1: Iterated adaptive filtering of white noise.

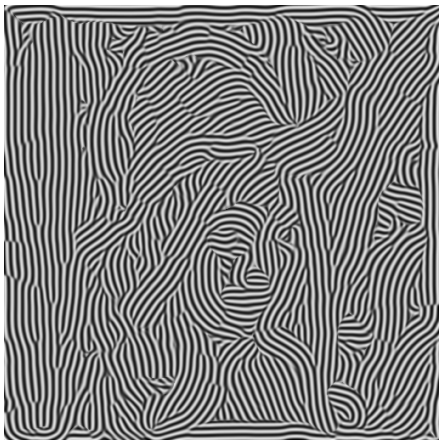
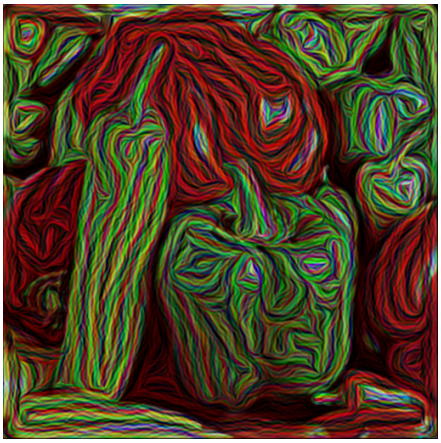


Figure J.2: Directed noise shaped to the orientation field of a real image.

(a) original image



(b) degraded image

Figure J.3: A more complex example of image degradation. The degraded image is available in color on the cover of [23].

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, June 1994.

[2] S. Aksoy, M. Ye, M. L. Schauf, M. Song, Y. Wang, R. M. Haralick, J. R. Parker, J. Pivovarov, D. Royko, C. Sun, and G. Farnebäck. Algorithm Performance Contest. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 4, pages 870–875, Barcelona, Spain, September 2000. IAPR.

[3] K. Andersson. Quality and motion estimation for image sequence coding. Lic. Thesis LiU-Tek-Lic-2002:01, Dept. Biomedical Engineering, Linköping University, SE-581 85 Linköping, Sweden, February 2002. Thesis No. 928, ISBN 91-7373-264-8.

[4] K. Andersson, P. Johansson, R. Forcheimer, and H. Knutsson. Backward-forward motion compensated prediction. In *Advanced Concepts for Intelligent Vision Systems (ACIVS'02)*, pages 260–267, Ghent, Belgium, September 2002.

[5] K. Andersson and H. Knutsson. Continuous normalized convolution. In *Proceedings of International Conference on Multimedia and Expo (ICME'02)*, pages 725–728, Lausanne, Switzerland, August 2002.

[6] M. Andersson, J. Wiklund, and H. Knutsson. Sequential Filter Trees for Efficient 2D 3D and 4D Orientation Estimation. Report LiTH-ISY-R-2070, ISY, SE-581 83 Linköping, Sweden, November 1998. URL: http://www.isy.liu.se/cvl/ScOut/TechRep/TechRep.html.

[7] A. Bab-Hadiashar and D. Suter. Robust optic flow computation. *International Journal of Computer Vision*, 29(1):59–77, August 1998.

[8] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *Int. J. of Computer Vision*, 12(1):43–77, 1994.

[9] J. Bigün. *Local Symmetry Features in Image Processing.* PhD thesis, Linköping University, Sweden, 1988. Dissertation No 179, ISBN 91-7870-334-4.

[10] J. Bigün and G. H. Granlund. Optimal Orientation Detection of Linear Symmetry. In *Proceedings of the IEEE First International Conference on Computer Vision*, pages 433–438, London, Great Britain, June 1987.

[11] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, Society for Industrial and Applied Mathematics, 1996.

[12] M. J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, Jan. 1996.

[13] M. J. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996.

[14] A.W. Bojanczyk and A. Lutoborski. Computation of the euler angles of a symmetric 3×3 matrix. *SIAM J. Matrix Anal. Appl.*, 12(1):41–48, January 1991.

[15] R. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, 2nd edition, 1986.

[16] P. J. Burt. Moment images, polynomial fit filters and the problem of surface interpolation. In *Proc. of Computer Vision and Pattern Recognition, Ann Arbor*. Computer Society Press, 1988.

[17] K. Daniilidis and V. Krüger. Optical flow computation in the log-polar plane. In *Proceedings 6th Int. Conf. on Computer Analysis of Images and Patterns*, pages 65–72. Springer-Verlag, Berlin, Germany, 1995.

[18] C. Dorf, editor. *The Electrical Engineering Handbook*. CRC Press LLC, 2000.

[19] F. Dufaux and F. Moscheni. Segmentation-based motion estimation for second generation video coding techniques. In L. Torres and M. Kunt, editors, *Video Coding: The Second Generation Approach*, chapter 6, pages 219–263. Kluwer Academic Publishers, 1996.

[20] L. Eldén. A Weighted Pseudoinverse, Generalized Singular Values, and Constrained Least Squares Problems. *BIT*, 22:487–502, 1982.

[21] G. Farnebäck. Motion-based Segmentation of Image Sequences. Master's thesis, Linköping University, SE-581 83 Linköping, Sweden, May 1996. LiTH-ISY-EX-1596.

[22] G. Farnebäck. Motion-based Segmentation of Image Sequences using Orientation Tensors. In *Proceedings of the SSAB Symposium on Image Analysis*, pages 31–35, Stockholm, March 1997. SSAB.

[23] G. Farnebäck. Spatial Domain Methods for Orientation and Velocity Estimation. Lic. Thesis LiU-Tek-Lic-1999:13, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 1999. Thesis No. 755, ISBN 91-7219-441-3.

[24] G. Farnebäck. A Unified Framework for Bases, Frames, Subspace Bases, and Subspace Frames. In *Proceedings of the 11th Scandinavian Conference on Image Analysis*, pages 341–349, Kangerlussuaq, Greenland, June 1999. SCIA. Also as Technical Report LiTH-ISY-R-2246.

[25] G. Farnebäck. Fast and Accurate Motion Estimation using Orientation Tensors and Parametric Motion Models. In *Proceedings of 15th International Conference on Pattern Recognition*, volume 1, pages 135–139, Barcelona, Spain, September 2000. IAPR.

[26] G. Farnebäck. Orientation Estimation Based on Weighted Projection onto Quadratic Polynomials. In B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2000: proceedings*, pages 89–96, Saarbrücken, November 2000.

[27] G. Farnebäck. Disparity Estimation from Local Polynomial Expansion. In *Proceedings of the SSAB Symposium on Image Analysis*, pages 77–80, Norrköping, March 2001. SSAB.

[28] G. Farnebäck. Very High Accuracy Velocity Estimation using Orientation Tensors, Parametric Motion, and Simultaneous Segmentation of the Motion Field. In *Proceedings of the Eighth IEEE International Conference on Computer Vision*, volume I, pages 171–177, Vancouver, Canada, July 2001.

[29] Gunnar Farnebäck and Klas Nordberg. Motion Detection in the WITAS Project. In *Proceedings SSAB02 Symposium on Image Analysis*, pages 99–102, Lund, March 2002. SSAB.

[30] H. G. Feichtinger and K. H. Gröchenig. Theory and practice of irregular sampling. In J. Benedetto and M. Frazier, editors, *Wavelets: Mathematics and Applications*, pages 305–363. CRC Press, 1994.

[31] D. J. Fleet and A. D. Jepson. Computation of Component Image Velocity from Local Phase Information. *Int. Journal of Computer Vision*, 5(1):77–104, 1990.

[32] James Foran. *Fundamentals of Real Analysis*. M. Dekker, New York, 1991. ISBN 0-8247-8453-7.

[33] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *ISRPS Intercommission Workshop*, pages 149–155, Interlaken, June 1987.

[34] Mats Gökstorp. *Depth Computation in Robot Vision*. PhD thesis, Linköping University. Sweden, SE-581 83 Linköping, Sweden, 1995. Dissertation No. 377, ISBN 91-7871-522-9.

[35] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, second edition, 1989.

[36] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1989.

[37] C. Gramkow. *2D and 3D Object Measurement for Control and Quality Assurance in the Industry*. PhD thesis, Technical University of Denmark, 1999.

[38] G. H. Granlund. In Search of a General Picture Processing Operator. *Computer Graphics and Image Processing*, 8(2):155–173, 1978.

[39] G. H. Granlund and H. Knutsson. Contrast of Structured and Homogenous Representations. In O. J. Braddick and A. C. Sleigh, editors, *Physical and Biological Processing of Images*, pages 282–303. Springer Verlag, Berlin, 1983.

[40] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[41] R. M. Haralick. Edge and region analysis for digital image data. *Computer Graphics and Image Processing*, 12(1):60–73, January 1980.

[42] R. M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(1):58–68, January 1984.

[43] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*, volume 1, chapter 8, The Facet Model. Addison-Wesley, 1992.

[44] R. M. Haralick and L. Watson. A facet model for image data. *Computer Graphics and Image Processing*, 15(2):113–129, February 1981.

[45] D. J. Heeger. Model for the extraction of image flow. *J. Opt. Soc. Am. A*, 4(8):1455–1471, 1987.

[46] Magnus Hemmendorff. *Motion Estimation and Compensation in Medical Imaging*. PhD thesis, Linköping University, Sweden, SE-581 85 Linköping, Sweden, July 2001. Dissertation No 703, ISBN 91-7373-060-2.

[47] M. Irani and P. Anandan. A unified approach to moving object detection in 2d and 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):577–589, June 1998.

[48] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, Seattle, Wa, June 1994.

[49] B. Jähne. Motion determination in space-time images. In *Image Processing III*, pages 147–152. SPIE Proceedings 1135, International Congress on Optical Science and Engineering, 1989.

[50] B. Jähne. Motion determination in space-time images. In O. Faugeras, editor, *Computer Vision-ECCV90*, pages 161–173. Springer-Verlag, 1990.

[51] B. Jähne. *Digital Image Processing: Concepts, Algorithms and Scientific Applications*. Springer Verlag, Berlin, Heidelberg, 1991.

[52] B. Jähne. *Spatio-Temporal Image Processing: Theory and Scientific Applications*. Springer Verlag, Berlin, Heidelberg, 1993. ISBN 3-540-57418-2.

[53] B. Jähne. *Practical Handbook on Image Processing for Scientific Applications*. CRC Press LLC, 1997. ISBN 0-8493-8906-2.

[54] B. Johansson. Multiscale Curvature Detection in Computer Vision. Lic. Thesis LiU-Tek-Lic-2001:14, Dept. EE, Linköping University, SE-581 83 Linköping, Sweden, March 2001. Thesis No. 877, ISBN 91-7219-999-7.

[55] Björn Johansson and Gunnar Farnebäck. A Theoretical Comparison of Different Orientation Tensors. In *Proceedings SSAB02 Symposium on Image Analysis*, pages 69–73, Lund, March 2002. SSAB.

[56] Björn Johansson and Gösta Granlund. Fast Selective Detection of Rotational Symmetries using Normalized Inhibition. In *Proceedings of the 6th European Conference on Computer Vision*, volume I, pages 871–887, Dublin, Ireland, June 2000.

[57] Björn Johansson, Hans Knutsson, and Gösta Granlund. Detecting Rotational Symmetries using Normalized Convolution. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 3, pages 500–504, Barcelona, Spain, September 2000. IAPR.

[58] F.de Jong, L.J. van Vliet, and P.P. Jonker. Gradient estimation in uncertain data. In *MVA'98 IAPR Workshop on Machine Vision Applications*, pages 144–147, Makuhari, Chiba, Japan, November 1998.

[59] S. X. Ju, M. J. Black, and A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proceedings CVPR'96*, pages 307–314. IEEE, 1996.

[60] T. Kailath. *Linear Systems*. Information and System Sciences Series. Prentice-Hall, Englewood Cliffs, N.J., 1980.

[61] J. Karlholm. *Local Signal Models for Image Sequence Analysis*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1998. Dissertation No 536, ISBN 91-7219-220-8.

[62] H. Knutsson. Representing Local Structure Using Tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland, June 1989. Report LiTH-ISY-I-1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.

[63] H. Knutsson and M. Andersson. Optimization of Sequential Filters. In *Proceedings of the SSAB Symposium on Image Analysis*, pages 87–90, Linköping, Sweden, March 1995. SSAB. LiTH-ISY-R-1797. URL: http://www.isy.liu.se/cvl/ScOut/ TechRep/TechRep.html.

[64] H. Knutsson and C-F. Westin. Normalized and Differential Convolution: Methods for Interpolation and Filtering of Incomplete and Uncertain Data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 515–523, New York City, USA, June 1993. IEEE.

[65] H. Knutsson, C-F. Westin, and G. H. Granlund. Local Multiscale Frequency and Bandwidth Estimation. In *Proceedings of IEEE International Conference on Image Processing*, pages 36–40, Austin, Texas, November 1994. IEEE.

[66] H. Knutsson, C-F. Westin, and C-J. Westelius. Filtering of Uncertain Irregularly Sampled Multidimensional Data. In *Twenty-seventh Asilomar Conf. on Signals, Systems & Computers*, pages 1301–1309, Pacific Grove, California, USA, November 1993. IEEE.

[67] J.J. Koenderink and A.J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[68] R. Kumar, P. Anandan, and K. Hanna. Direct recovery of shape from multiple views: a parallax based approach. In *Proceedings of 12th ICPR*, pages 685–688, October 1994.

[69] S.-H. Lai and B. C. Vemuri. Reliable and efficient computation of optical flow. *International Journal of Computer Vision*, 29(2):87–105, August/September 1998.

[70] H. Liu, T-H. Hong, M. Herman, and R. Chellappa. A general motion model and spatio-temporal filters for computing optical flow. *International Journal of Computer Vision*, 22(2):141–172, 1997.

[71] B. Lucas and T. Kanade. An Iterative Image Registration Technique with Applications to Stereo Vision. In *Proc. Darpa IU Workshop*, pages 121–130, 1981.

[72] L. Massone, G. Sandini, and V. Tagliasco. "Form-invariant" topological mapping strategy for 2d shape recognition. *Computer Vision, Graphics, and Image Processing*, 30(2):169–188, May 1985.

[73] E. Mémin and P. Pérez. Hierarchical estimation and segmentation of dense motion fields. *International Journal of Computer Vision*, 46(2):129–155, February 2002.

[74] K. Nordberg. *Signal Representation and Processing using Operator Groups*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1995. Dissertation No 366, ISBN 91-7871-476-1.

[75] W. Rudin. *Real and Complex Analysis*. McGraw-Hill, 3rd edition, 1987. ISBN 0-07-054234-1.

[76] H. S. Sawhney. 3d geometry from planar parallax. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 929–934, June 1994.

[77] H. R. Schwarz. *Numerical Analysis: A Comprehensive Introduction*. John Wiley & Sons Ltd., 1989. ISBN 0-471-92064-9.

[78] A. Shashua and N. Navab. Relative affine structure: Theory and application to 3d reconstruction from perspective views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 483–489, June 1994.

[79] O. K. Smith. Eigenvalues of a symmetric 3×3 matrix. *Communications of the ACM*, 4(4):168, 1961.

[80] G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, 1986. ISBN 0-9614088-0-4.

[81] R. Szeliski and J. Coughlan. Hierarchical spline-based image registration. In *Proc. IEEE Conference on Computer Vision Pattern Recognition*, pages 194–201, Seattle, Washington, 1994.

[82] M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):401–410, April 1993.

[83] A. Torchinsky. *Real Variables*. Addison-Wesley, 1988. ISBN 0-201-15675-X.

[84] M. Ulvklo, G. H. Granlund, and H. Knutsson. Texture Gradient in Sparse Texture Fields. In *Proceedings of the 9th Scandinavian Conference on Image Analysis*, pages 885–894, Uppsala, Sweden, June 1995. SCIA.

[85] M. Ulvklo, G. H. Granlund, and H. Knutsson. Adaptive Reconstruction Using Multiple Views. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 47–52, Tucson, Arizona, USA, April 1998. IEEE. LiTH-ISY-R-2046.

[86] M. Ulvklo, H. Knutsson, and G. H. Granlund. Depth Segmentation and Occluded Scene Reconstruction using Ego-motion. In *Proceedings of the SPIE Conference on Visual Information Processing*, pages 112–123, Orlando, Florida, USA, April 1998. SPIE.

[87] S. Uras, F. Girosi, A. Verri, and V. Torre. A computational approach to motion perception. *Biological Cybernetics*, pages 79–97, 1988.

[88] B. L. v. d. Waerden. *Algebra*, volume 1. Springer-Verlag, Berlin, Göttingen, Heidelberg, 5 edition, 1960.

[89] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625–638, September 1994.

[90] J. Y. A. Wang and E. H. Adelson. Spatio-temporal segmentation of video data. In *Proceedings of the SPIE Conference: Image and Video Processing II*, volume 2182, pages 120–131, San Jose, California, February 1994.

[91] J. Y. A. Wang, E. H. Adelson, and U. Y. Desai. Applying mid-level vision techniques for video data compression and manipulation. In *Proceedings of the SPIE: Digital Video Compression on Personal Computers: Algorithms and Technologies*, volume 2187, pages 116–127, San Jose, California, February 1994.

[92] C. F. R. Weiman and G. Chaikin. Logarithmic spiral grids for image processing and display. *Computer Graphics and Image Processing*, 11:197–226, 1979.

[93] C-J. Westelius. *Focus of Attention and Gaze Control for Robot Vision*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1995. Dissertation No 379, ISBN 91-7871-530-X.

[94] C-F. Westin. *A Tensor Framework for Multidimensional Signal Processing*. PhD thesis, Linköping University, Sweden, SE-581 83 Linköping, Sweden, 1994. Dissertation No 348, ISBN 91-7871-421-4.

[95] C-F. Westin, K. Nordberg, and H. Knutsson. On the Equivalence of Normalized Convolution and Normalized Differential Convolution. In *Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing*, pages 457–460, Adelaide, Australia, April 1994. IEEE.

[96] C-F. Westin, C-J. Westelius, H. Knutsson, and G. Granlund. Attention Control for Robot Vision. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 726–733, San Francisco, California, June 1996. IEEE Computer Society Press.

[97] R. Wilson and H. Knutsson. Uncertainty and inference in the visual system. *IEEE Transactions on Systems, Man and Cybernetics*, 18(2):305–312, March/April 1988.

[98] R. Wilson and H. Knutsson. Seeing Things. Report LiTH-ISY-R-1468, Computer Vision Laboratory, SE-581 83 Linköping, Sweden, 1993.

[99] WITAS web page.
`http://www.ida.liu.se/ext/witas/eng.html`.

[100] S. Xu. *Motion and Optical Flow in Computer Vision*. PhD thesis, Linköping University. Sweden, SE-581 83 Linköping, Sweden, 1996. Dissertation No. 428, ISBN 91-7871-335-3.

[101] M. Ye and R. M. Haralick. Image flow estimation using facet model and covariance propagation. In M. Cheriet and Y. H. Yang, editors, *Vision Interface: Real World Applications of Computer Vision*, volume 35 of *Series in Machine Perception and Artificial Intelligence*, pages 209–241. World Scientific Pub Co., 2000.