Hierarchical Curvature Estimation in Computer Vision

Håkan Bärman

Department of Electrical Engineering
Linköping University, S-581 83 Linköping, Sweden

Linköping 1991
Abstract

This thesis concerns the estimation and description of curvature for computer vision applications. Different types of multi-dimensional data are considered: images (2D); volumes (3D); time sequences of images (3D); and time sequences of volumes (4D).

The methods are based on local Fourier domain models and use local operations such as filtering. A hierarchical approach is used. Firstly, the local orientation is estimated and represented with a vector field equivalent description. Secondly, the local curvature is estimated from the orientation description. The curvature algorithms are closely related to the orientation estimation algorithms and the methods as a whole give a unified approach to the estimation and description of orientation and curvature. In addition, the methodology avoids thresholding and premature decision making.

Results on both synthetic and real world data are presented to illustrate the algorithms performance with respect to accuracy and noise insensitivity. Examples illustrating the use of the curvature estimates for tasks such as image enhancement are also included.
Acknowledgements

I would like to express my gratitude to those who have helped me in pursuing this work:

To my supervisor, Gösta H. Granlund, professor of Computer Vision at LiTH. He initiated the ideas behind the algorithms and provided excellent guidance as well as encouragement throughout the work.

To Dr. Hans Knutsson, for being a constant source of comments, explanations, ideas, modifications, tests, etc.

To Johan Wiklund, who wrote the source code for the 3D and 4D image handling package including the support for 3D and 4D convolutions. My thesis would have been considerably thinner without this package.

To all the past and present members of the Computer Vision Laboratory for their numerous contributions to this work, the friendly and stimulating atmosphere, and the proof-reading of this manuscript in its various forms. All mis-spellings etc. left in this document are probably due to my last minute changes. Dr. Andrew Calway made many suggestions which improved the readability of this thesis.

To the Swedish National Board for Technical Development for its financial support of this work.

Finally to Sofia for her patience, love and support.
# Contents

1 Introduction ........................................ 7
   1.1 A Unified Approach to Curvature Estimation .......... 9
   1.2 Overview ........................................ 10

I 2D — Image Processing .................................. 13

2 Curvature Estimation and Description Using Vector Field Filtering 15
   2.1 Properties of Orientation Vector Images ........... 15
   2.2 Curvature Algorithms .............................. 22
      2.2.1 Curvature Direction ......................... 23
      2.2.2 Curved/Straight Ratio ....................... 31
      2.2.3 Curvature Magnitude ......................... 33
   2.3 Results ........................................... 35
      2.3.1 Synthetic Vector Images .................... 35
      2.3.2 Synthetic Grey Level Images ............... 37
      2.3.3 Natural Images ............................. 38
   2.4 Conclusions ....................................... 41
   2.A Magnitude Invariance of Quasi-Quadrature Filters 41
   2.B Using Quadrature Filters in Equation (2.18) ........ 43
   2.C The Filter Interpolation Procedure ............. 43
   2.D Design of Filters for the ‘Standard’ Algorithm (eq. (1.2)) 44

3 A Filtering Strategy for Orientation and Curvature Description 46
   3.1 Background ...................................... 46
   3.2 Normalization .................................... 47
   3.3 Performance Measures ............................ 47
   3.4 Results .......................................... 48
   3.5 Summary ......................................... 51
   3.A Test Patterns and Error Measures ................ 54

4 Application Examples: Texture Analysis and Image Enhancement 56
   4.1 Texture Analysis ................................... 56
   4.2 Image Enhancement: Grey Level Images ............ 59
      4.2.1 The Knutsson, Wilson and Granlund Algorithm 61
      4.2.2 Curvature Modifications to the Algorithm ...... 65
   4.3 Image Enhancement: Vector Images ................ 69
   4.4 Conclusion ......................................... 71
II 3D — Volume and Time Sequence Processing

5 Estimation, Representation and Visualization of 3D Orientation
5.1 Using Tensors to Represent 2D Orientation
5.2 3D Orientation Estimation and Representation
5.3 Accuracy of the Orientation Estimate
5.4 Visualization of the Orientation Tensor
  5.4.1 Visualization of an Arbitrary Orientation Plane
5.5 Summary
5.A The Mapping from 2D Orientation Tensor to ‘Double Angle’ Vector
5.B How to Implement the Orientation Algorithm
5.C Arbitrary Orientation Plane Visualization

6 The Principal Direction Algorithm
6.1 The Problem
  6.1.1 Related Work
6.2 The Representation
6.3 The Model
6.4 The Algorithm
  6.4.1 Estimation of the Centre of Gravity in the Local Fourier Spectra
  6.4.2 Extraction of the Principal Directions
6.5 Improvements of the Principal Direction Algorithm
  6.5.1 Averaging
  6.5.2 Orientation Consistency
  6.5.3 Eigenvalue Consistency
6.6 Local Shape Analysis
  6.6.1 Umbilics
  6.6.2 Gaussian Curvature
6.7 Examples
6.8 Summary
6.A The Continuous Representation of Principal Directions of Curvature
6.B The Complex-Valued Orientation Representation
6.C Projection of 3D Normal Vectors into 2D Vectors
6.D Generalization of the Parabolic Model to Elliptic and Hyperbolic
  Neighbourhoods
6.E Estimation of Centre of Gravity
6.F Obtaining Six Complex Filter Responses by Four Scalar Filters
6.G Fourier Domain Directions of the Filter Set
6.H Implementation of the Algorithm
6.I Rotation Invariance
6.J Proof of eq. (6.44)
6.K Proof of eq. (6.45)
Chapter 1

Introduction

The main theme of this thesis is the estimation and description of curvature for computer vision applications. It is widely accepted that curvature is a relevant property of images and thus useful for computer vision tasks. Moreover, the human visual system offers additional motivation for adopting curvature as an image feature, since there are many indications of curvature detection being a mechanism in human perception. A well-known example of this was the demonstration by Attneave [5] that the approximation of the silhouette of a cat by straight lines ending in curvature maxima resulted in a figure which was still recognizable as a cat. Consequently curvature estimation forms an important tool in computer vision. Its use extends from guiding the selection of points in, for example, both the polygonal [80] and the polynomial [18] approximation of curves, to matching in the estimation of the three-dimensional orientation of objects from two-dimensional images [36].

Curvature is a relatively simple concept which is readily appreciated from fig. 1.1. Stated simply, curvature describes the changes in orientation of a curve and is commonly defined with the use of the curve's tangent, i.e.

$$\kappa = \left| \frac{dt}{ds} \right| = \frac{d\theta}{ds} \tag{1.1}$$

where \( t \) is a unit tangent vector, \( \theta \) is the tangent vector's angle and \( s \) denotes the arc length. This can in terms of an image \( f(\xi_1, \xi_2) \) be written as

$$\kappa = \frac{1}{(\frac{\partial^2 f}{\partial \xi_1^2})^2 + (\frac{\partial^2 f}{\partial \xi_2^2})^2} \left[ \frac{\partial^2 f}{\partial \xi_1 \partial \xi_2} \right] \left[ \frac{\partial^2 f}{\partial \xi_1^2} \frac{\partial^2 f}{\partial \xi_2} - 2 \frac{\partial f}{\partial \xi_1} \frac{\partial f}{\partial \xi_2} \right] \left[ \frac{\partial^2 f}{\partial \xi_1 \partial \xi_2} \right] \tag{1.2}$$

The term 'curvature' does here, for 2D images, refer to a feature of the curves in the image. It is, however, worth pointing out that if the image is interpreted as a surface, i.e. the pixel values represent 'height values', curvature would then instead be a feature of that surface.

The need for reliable estimation of curvature in computer vision has led to considerable theoretical and practical investigations. Koenderink [69, 68] was one of the pioneers introducing the concepts of differential geometry into the field of Computer Vision. Estimation schemes have traditionally been based on line and/or edge detection followed by thresholding before the actual curvature estimation takes place.
Methods in this category have been based on $k$-curvature [81, 95], $\psi$-s curves [7] and polynomial approximation [72, 81]. However, a significant drawback of these approaches concerns the use of thresholding, which tends to introduce some serious difficulties when considering real application problems, e.g:

- If noise is present, which is generally the case, it is hard or even impossible to select a proper threshold to discriminate the curve from the background.
- The conversion to a binary image introduces additional quantization noise since angular changes only occur in multiples of $\pi/4$.
- Small changes in the threshold level may cause dramatically different curvature estimates.

Parent and Zucker [78, 79] have recognized many of these problems and modified the well-known relaxation scheme of Rosenfeld, Hummel and Zucker [86] to produce a procedure for curve detection which takes trace, tangent and curvature information into account. This iterative method discriminates between a few different curvature classes (typically four, with the fourth one being equivalent to a straight line), which is enough for acceptable curve detection on relatively complex images. Thresholding was used by Asada and Brady [4] although they incorporated a scale-space approach (Witkin [102]) which led to a more robust algorithm. A similar strategy was also utilized by Leymarie and Levine [70]. The curvature formula given in eq. (1.2) can also be used directly for curvature estimation (e.g. Machuca and Phillips [71] or Kass et al [57]) and has the advantage of not requiring thresholding. However, this direct method tends to be noise sensitive.
1.1 A Unified Approach to Curvature Estimation

The curvature estimation and description algorithms described in this thesis are in many respects different to other methods. The most significant is that the method gives a unified approach to the estimation and description of orientation and curvature. In addition, the methodology avoids thresholding and premature decision making.

The basic outline of the method is illustrated in fig. 1.2. The input data to the curvature algorithm consists of an orientation description of the image, i.e., a complex-valued image in which each point represents an estimate of the dominant orientation in a local region about the point, viz

\[ f_1(\xi_1, \xi_2) = a + ib = re^{i\phi} \]  

(1.3)

where the magnitude \( r \) is a 'certainty measure' that the region has a dominant orientation and \( \phi \) is the estimate of that orientation. Note that the representation used is based on the 'double angle' representation, which gives some essential benefits such as continuity of the representation of line orientation and ensuring that orientation averaging is a meaningful operation. The scheme shown in fig. 1.2 is based primarily on the hierarchical methodology for Computer Vision which has evolved at the Computer Vision Laboratory, Linköping University, during the last decade [19, 20, 21, 40, 45, 61, 64]. An important aspect of the approach is that information is represented in the same way at every level of the hierarchy, i.e., the symbols look similar at every level, while the meaning of such a symbol varies depending on which level it is located. More detailed discussions and arguments for using such a data representation are given later in the thesis.

One way of obtaining the orientation image is by convolution with a set of orientation selective quadrature filters. An example of such a filter is given in fig. 1.3.
Figure 1.3: A quadrature filter in 22.5° direction

where the vectors represent the complex-valued coefficients of the filter. Such an
approach to orientation estimation has been successfully developed previously by
Knutsson [64]. The work in this thesis is based on the observation that curvature is
reflected in the orientation image as the speed of angular change (fig. 1.1). Noting
that similar properties are exhibited by the filter in fig. 1.3 then immediately poses
the question: ‘Would it not be possible to estimate curvature by correlating the ori-
entation vector image with sets of quadrature filters, thus yielding a consistent and
unified approach to curvature estimation?’. This thesis will give a positive answer
to that question.

1.2 Overview

The thesis is organized into three different parts, one each for two-, three- and
four-dimensional images. The algorithms are nevertheless founded on the same
ideas and use similar strategies. The essence is the general curvature estimation
method outlined above which is not limited to any particular type of application.
A certain amount of the material in the different parts necessarily overlap to some
extent, and an alternative organization might have been to combine the parts into
one grand theory of curvature estimation and description in N-dimensional images.
Curvature is, however, a much more simple entity in 2D than in 3D (and 4D)
images and the grand theory approach would have made the presentation of the 2D
material unnecessary complicated. It therefore seemed more appropriate to present
the material separately.

The first part describes the 2D curvature algorithms and includes a detailed
description of the orientation estimation algorithm, since this method is essential
for understanding of the curvature algorithms. Examples of both bottom-up and
top-down (feedback) processing within the hierarchical structure are included.

The second part generalizes the algorithm to 3D data in the form of volumes and
time sequences. Included is a description of the orientation estimation and representation method extended to 3D with the appropriate additions and modifications.

Finally, the scheme is extended to 4D data in part III. The thesis concludes with a summary including a short comparison with known or apparent processing within the human visual system, and a discussion of future work.

Proofs and other technical details are collected at the end of each chapter in appendix-fashion with letters used to number these sections (e.g., 3.A) with a real appendix at the end of the thesis, which contains all the colour images of the thesis.
Part I

2D — Image Processing

They're just transformations
Variations
Alternations
Deviations
You know Mother Nature rules us all

"TRANSFORMATIONS"— Nona Hendryx
Chapter 2

Curvature Estimation and Description Using Vector Field Filtering

This chapter presents a new method for the detection and estimation of curvature. The algorithm is implemented in a hierarchical feature pyramid structure. Curvature is handled at the second level of the pyramid with a vector field description of the orientation of the image as input. This complex image is convolved with typically four (complex-valued) filters. The filter outputs are combined into a description of the curvature, including both magnitude and direction. The procedure resembles in many ways the algorithms for the first level of the feature pyramid and is a natural extension of these. Some comparisons with other algorithms have been carried out, and the results indicate that the methodology presented here has a number of important advantages over other methods.

The chapter is divided into three sections. First, the initial step of achieving an orientation image is described together with a discussion of the frequency domain properties of curvature in an orientation image. Next, the second level curvature algorithms, handling the orientation image, are presented and discussed in the context of their counterparts (orientation estimation etc) at the first level of the pyramid. Finally, a presentation and evaluation of the experimental results are given.

2.1 Properties of Orientation Vector Images

The production of an orientation description is the first of two steps from grey level image to curvature description (fig. 1.2). This section is an account of how the orientation information is obtained and represented and how it relates to the curvature properties of the original image.

A grey level image can be represented by a function $f_0(\xi_1, \xi_2)$, where $\xi_1$ and $\xi_2$ are the spatial coordinates and the function value represents the grey level at that image point. It is well known that the image function $f_0(\xi_1, \xi_2)$ can be transformed into the frequency domain by the Fourier transform. The corresponding function is denoted $F_0(u_1, u_2)$, where $u_1$ and $u_2$ are the frequency coordinates (An excellent account of the Fourier transform is given in Bracewell [27]).
Since the objective is to derive a local description of the image, it is natural to pay attention to Fourier transforms of small neighbourhoods and not to the entire image. A local neighbourhood in the image can be said to be one-dimensional if the energy in the Fourier domain is concentrated along a line through the origin oriented at an angle $\phi$. Neighbourhoods in the spatial domain with gradients in the $\phi$ direction give rise to such Fourier transforms. This property is used to compile the orientation description, a data representation based on a number of observations.

- Data compression is essential to provide manageable information. There is little use in storing extracted data in a way that makes handling of the information as difficult as deriving it from the original image. On the other hand it is important to ensure that no valuable information is lost.

- Events are not likely to occur simultaneously. That is, two intersecting perpendicular lines are rare events from a statistical point of view. One or a few parameters are thus adequate to describe most neighbourhoods.

- The representation shall also be able to describe the accuracy of the description. That is, when the two lines mentioned above actually cross, the representation shall indicate the failure in representing one-dimensional orientation and not try to describe that 'double-event'.

This leads to the complex-valued orientation description proposed by Granlund [40] and introduced in chapter 1. The argument represents the orientation with 'double-angle', see fig. 2.1, and the magnitude is a measurement of the certainty of the estimated orientation. The orientation description refers to the dominant orientation of the neighbourhood, and is consequently not able to give an adequate description of 'double-event' neighbourhoods. Such a neighbourhood is represented with a vector of zero or low magnitude, which indicates a breakdown of the 'one dominant orientation' model. Note that at this stage no regard is taken to whether the neighbourhood is constituted by a line or an edge (Discrimination between lines and edges is considered by Knutsson [64] and Haglund [42]).

Complex-valued images are also used to represent other kinds of information, always with the magnitude as certainty estimate and the argument representing the actual value.

The local orientation estimation algorithm used in this work was developed by Knutsson [64] and works like the following. The estimate is achieved by combining the outputs from a number of filters, at least three, where each filter collects energy from a partition of the Fourier domain. Since the image $f_0(\xi_1, \xi_2)$ is real-valued, its Fourier transform $F_0(u_1, u_2)$ is Hermitian, i.e., the real part is an even function and the imaginary part is an odd function. This implies that it is sufficient to use filters spanning half of the Fourier domain.

The filters that are used are quadrature filters which only take up energy in one half-plane of the Fourier domain [27, 64]. By letting $H_k(u_1, u_2)$, the Fourier representation of filter $k$, be real-valued, the spatial representation of the filter, $h_k(\xi_1, \xi_2)$, will in turn become Hermitian with the even real part as line detector and the odd imaginary part as edge detector.

\[ The original algorithm by Granlund [40] used Gabor filters instead of quadrature filters. \]
A way of constructing such a filter is

\[
H_k(u_1, u_2) = H_{ke}(u_1, u_2) + H_{ko}(u_1, u_2) \tag{2.1}
\]

\[
H_{ke}(u_1, u_2) = H_{ke}(-u_1, -u_2) \tag{2.2}
\]

\[
H_{ko}(u_1, u_2) = \text{sign} \left[ \cos(\varphi - \phi_k) \right] H_{ke}(u_1, u_2) \tag{2.3}
\]

where \( \varphi = \arg(u_1 + iu_2) \) and \( \phi_k \) is the direction of the filter. This gives in the spatial domain

\[
h_k(\xi_1, \xi_2) = h_{ke}(\xi_1, \xi_2) + ih_{ko}(\xi_1, \xi_2) \tag{2.4}
\]

The even part of the filter \( h_k \) is thus purely real, and the odd part is purely imaginary. Spatial and frequency responses of a quadrature filter and its real and imaginary components are illustrated in fig. 2.2-2.5.

By spreading the filters equally over a half-plane of the frequency domain, the dominant orientation can then be achieved by vector addition

\[
f_1(\xi_1, \xi_2) = \sum_{k=1}^{K} q_k(\xi_1, \xi_2) e^{i\phi_k} \tag{2.5}
\]

where \( K \) is the number of filters and \( q_k(\xi_1, \xi_2) \) denotes the magnitude of the filter output,

\[
q_k = \sqrt{q_{ke}^2 + q_{ko}^2} \tag{2.6}
\]

with \( q_{ke} \) being the filter output of the even filter and \( q_{ko} \) being the filter output of the odd filter. The doubling of \( \phi_k \) in eq. (2.5) results in the double angle representation displayed in fig. 2.1. This strategy gives high magnitude, or certainty, to the orientation estimate when the neighbourhood is close to one-dimensional. 'Double-event'-type neighbourhoods will give comparatively lower magnitude since the two components will then be opposite directed in eq. (2.5) and tend to cancel out.
Figure 2.2: The spatial response of a quadrature filter. Top: real part. Bottom: imaginary part

Figure 2.3: The frequency response of the complex filter in fig. 2.2. Only the real part is displayed (the imaginary part is $\approx 0$)
Figure 2.4: The frequency response of the real component of the filter in fig. 2.2. Only the real part is displayed (the imaginary part is $\approx 0$)

Figure 2.5: The frequency response of the imaginary component of the filter in fig. 2.2. Only the real part is displayed (the imaginary part is $\approx 0$)
The magnitude of the vector summation should be rotation invariant, i.e., not dependent on the value of the orientation. This requirement is fulfilled if the function $H_{ke}(u_1, u_2)$ defining the frequency response of the quadrature filter is chosen as

$$H_{ke}(u_1, u_2) = W(\rho) \cos^2(\varphi - \phi_k) \tag{2.7}$$

where $\rho = \sqrt{u_1^2 + u_2^2}$ and $W$ is a radial weighting function controlling the frequency characteristics of the filter.

It is now appropriate to examine how the curvature in a 2D grey level image is transferred into the orientation image $f_1(\xi_1, \xi_2)$. The orientation image describes orientation in terms of local neighbourhoods, and not as curves or curve tangents. Thus the definition (1.1) is not directly applicable.

Of more relevance is the observation that curvature is proportional to 'the rate of change' of orientation along a curve and that this is reflected in the argument of the orientation image, i.e., $f = 2\phi = 2\theta + \pi$ (see fig. 1.1). Note that a large magnitude $r$ of the orientation estimate indicates that only one curve or orientation is present.

The orientation image function $f_1(\xi_1, \xi_2)$ can be rewritten as

$$f_1(\xi_1, \xi_2) = r(\xi_1, \xi_2)e^{i\varphi(\xi_1, \xi_2)} \tag{2.8}$$

thus making explicit that $r$ and $\theta$ are functions of the spatial coordinates. The quantities of interest are then $\varphi(\xi_1, \xi_2)$ and its gradient,

$$\nabla \varphi(\xi_1, \xi_2) = \frac{\partial \varphi(\xi_1, \xi_2)}{\partial \xi_1} \xi_1 + \frac{\partial \varphi(\xi_1, \xi_2)}{\partial \xi_2} \xi_2 \tag{2.9}$$

but only on the curve, i.e., at positions with well defined orientation values. This information is contained in the magnitude $r(\xi_1, \xi_2)$ and suggests the following method to detect, estimate and describe the main local curvature in the orientation image:

- use the magnitudes in a neighbourhood so that only high certainty points (pixels on the curve) have a significant effect on the computation.
- estimate the gradient of $\varphi(\xi_1, \xi_2)$ for those points.
- represent the achieved information in a compact and easily manageable way.

The motivation for the last item on the list is the same as for the case of local orientation description, where one of the base components is the hypothesis of one-dimensionality. 'One-dimensionality' refers to that it is generally true that a small neighbourhood at most contains a single event. Features of the neighbourhood can thus be described with relatively few parameters, e.g., one value to describe the local orientation.

This motivates the postulate that it is not likely that a small neighbourhood contains more than one type of curvature. This constraint makes it possible to express the angular function as

$$\varphi(\xi_1, \xi_2) = 2\pi(a\xi_1 + b\xi_2) \tag{2.10}$$
where $2\pi \sqrt{a^2 + b^2}$ is the magnitude of the curvature and \( \arg(a + ib) \) is the direction of the curve's tangent from which the curvature originates. The factor $2\pi$ in eq. (2.10) is added to simplify the forthcoming equation (2.11), but can of course be excluded. Information of the tangent may seem to be, apart from a sign, redundant information since tangent and gradient are perpendicular in 'clean' single-curve neighbourhoods. However, this is not necessarily the case for estimates in noisy neighbourhoods. Fig. 2.6 gives an example of a neighbourhood with the angular function given by eq. (2.10).

It is now possible to formulate a more general definition of curvature in terms of an amount of change, not of tangent angle of a curve, but of directional change in a vector field. Note that, however, this does not conflict with the earlier and more common definition (1.1), since the orientation algorithm ensures that the vector field is produced by one curve.

With the approximation of $\varphi(\xi_1, \xi_2)$ in eq. (2.10), it is fairly straightforward to go to the frequency domain and study how curvature is reflected there, viz

$$F_1(u_1, u_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} r(\xi_1, \xi_2) e^{2\pi i(u_1 \xi_1 + u_2 \xi_2)} e^{-2\pi i(u_1 \xi_1 + u_2 \xi_2)} d\xi_1 d\xi_2 = \hat{R}(u_1 - a, u_2 - b)$$

(2.11)

This shows that curvature is reflected in the Fourier domain as a shift. Furthermore,
Figure 2.7: a) The neighbourhood. b) Transform of orientation magnitude. c) Transform of orientation estimate. d) Transform after multiplication of the angles by four.

$R(u_1, u_2)$ will be Hermitian since $r(\xi_1, \xi_2)$ is real-valued.

The relevance of this equation is illustrated in fig. 2.7, which shows the Fourier transform of a neighbourhood of orientation vectors describing a circle. The white cross indicates the origin $u_1 = u_2 = 0$. The upper right transform is derived from the neighbourhood with all arguments set to zero, and would thus correspond to $R(u_1, u_2)$. The lower left sub-image is the ordinary transform. A trick is demonstrated in the lower right: before the transform is produced, all angles have been multiplied by four, and, as can be seen, this amplifies the offset due to curvature. This trick emphasizes the shift and can be used as preprocessing of the orientation data before the curvature estimation algorithms are used. Also observe that the linear orientation of $r(\xi_1, \xi_2)$ is orthogonal to the curvature direction of $\varphi(\xi_1, \xi_2)$.

This implies that curvature estimation is analogous to shift estimation in local Fourier transforms of the orientation image. The next section describes the design of such algorithms.

### 2.2 Curvature Algorithms

The curvature estimation is a two-step procedure (fig. 1.2). The preceding section described the first step of obtaining an orientation description of the image. This section describes the second step of the curvature description. The orientation description of the preceding section serves as input, and three different measures are derived to describe the curvature: direction (of the tangent), curved/straight ra-
2.2.1 Curvature Direction

The procedure of extracting the local curvature direction from the orientation data shares a great deal of detail with the algorithm for extraction of local orientation from the grey level data. The local orientation algorithm in the preceding section used properties of the Fourier domain such as the input being real-valued, a fact which is exploited in the use of quadrature filters. The structure of the local curvature algorithm have many points of similarity with the orientation algorithm, although the input data is now complex-valued.

The filters must span the entire Fourier domain since the Fourier transform of the complex-valued input data does not have the Hermitian property. A generalization of orientation estimation to complex images is given before introducing the curvature direction algorithm. The local orientation in a complex-valued neighbourhood is estimated by

\[ f'_2(\xi_1, \xi_2) = \sum_{k=1}^{2K} q_k(\xi_1, \xi_2)e^{i2\phi_k} \]  

(2.12)

where \( q_k \) now denotes the magnitude of the filter output from convolution of \( f_1(\xi_1, \xi_2) \) and not \( f_0(\xi_1, \xi_2) \) as in eq. (2.5). The number of filters is in eq. (2.12) written as \( 2K \) to emphasize that all of the Fourier domain is covered with filters and not only a half-plane (but \( K \) is used in all forthcoming equations). Let the filter directions \( \phi_k \) be the same as in eq. (2.5). The additional filters \( h_{K+1} \ldots h_{2K} \) are defined as

\[ h_{K+k}(\xi_1, \xi_2) = h_k^*(\xi_1, \xi_2) \]  

(2.13)

where \( * \) denotes complex conjugate and \( 2\phi_{K+k} = 2\phi_k \). It is easy to see that for filters designed according to eq. (2.1) – (2.4), the transform of the complex conjugate filter will yield a rotation of \( \pi \) compared to the original filter’s transform, since a change of sign in the odd part will make the even/odd cancellation take place at the other side of the Fourier domain (see fig. 2.4 and 2.5). The additional use of complex conjugate filters will consequently cover the whole frequency domain.

It may not be intuitively apparent what the orientation estimate obtained by eq. (2.12) corresponds to in the input image \( f_0(\xi_1, \xi_2) \). Isolated lines and edges will, however, have more or less identical orientation estimates in \( f_1(\xi_1, \xi_2) \) (eq. (2.5)) and \( f_2(\xi_1, \xi_2) \). The reason for this is that lines and edges in \( f_0(\xi_1, \xi_2) \) are preserved as lines in the magnitude part of \( f_1(\xi_1, \xi_2) \). The vector orientation algorithm (eq. (2.12)) does in fact more than the ‘scalar’ orientation algorithm (eq. (2.5)), since the former also takes angular changes into account. The difference is marginal on ‘normal’ images (whatever that is!), and the method is far from being a curvature descriptor.

The purpose is, however, to estimate curvature and not orientation. The relationship between spatial curvature and frequency domain shift, established in eq. (2.11), is essential together with the observation that the centre of gravity in the amplitude spectrum \( |F(u_1, u_2)| \) of a real-valued image is located in the origin. The latter follows from the Hermitian property. To calculate the shift, it is thus sufficient to estimate the centre of gravity of the neighbourhood’s amplitude spectrum and the curvature direction is calculated by estimation of the direction to the centre of gravity.
There exist well-known formulae to determine the centre of gravity of \( f(\xi_1, \xi_2) \) from \( F(u_1, u_2) \), and vice versa. The centres of gravity for \( F(u_1, u_2) \) and \( |F(u_1, u_2)| \) coincide, provided that there are no sign changes in the frequency domain. There is unfortunately nothing that ensures that condition. Another strategy is therefore proposed.

It is sufficient for the scalar orientation algorithm to estimate in a half-plane of the frequency domain. The orientation algorithm for complex-valued input data has filters in the entire domain, although the energies from the filters are mapped as if the neighbourhood was real-valued. The curvature estimation also requires filters of the whole domain, with the one difference being the mapping of filter energy

\[
f_2(\xi_1, \xi_2) = \sum_{k=1}^{K} q_k(\xi_1, \xi_2) e^{i\phi_k}
\]  

(2.14)

Observe that \( \phi_k \) no longer is doubled as in eq. (2.5) and (2.12). The double angle used in the orientation vector sum is thus in the context of curvature replaced by a 'single angle'. It may not be obvious that this method yields the direction of the shift, which is equivalent to the curvature direction, but a careful examination of the figures 2.7 and 2.8 together with eq. (2.14) should help. Fig. 2.8 contains an example in stylized form. Here four filters are used, with each of the filters concentrated in one partition of the Fourier domain (the exact shape of the filters is discussed later).

The location of the centre of gravity is indicated in fig. 2.8b and the magnitudes of the filter outputs in fig. 2.8c are shown as vectors directed in the corresponding filter directions. The rightmost vector in fig. 2.8 is obtained by eq. (2.14). The three equations (2.5), (2.12) and (2.14) are illustrated in fig. 2.9.

The curvature direction representation of a circle is exemplified in fig. 2.10. Note that both orientation and direction values are represented with complex numbers and that the curvature direction vectors coincide with the tangent direction. It is customary to assign the curvature value to a vector, the curvature vector, directed at the centre of the circle [90]. The curvature direction vectors of eq. (2.14) can, if one prefers such a representation, easily be modified by adding \( \pi/2 \) to the argument of the vector. This can be verified by inspection of fig. 2.10.

The algorithm adapts well to the proposal ending chapter 1 since the direction information for the image point \((\xi_1, \xi_2)\) is contained in the argument \( \varphi \) of \( f_2(\xi_1, \xi_2) \). However, how could the magnitude \( |f_2(\xi_1, \xi_2)| \) be interpreted?
Filter Directions in the Fourier Domain

Grey Level Orientation

Vector Orientation

Curvature Direction

Filter Directions in Vector Summation

Figure 2.9: The three different vector sums for $2K = 8$. The upper part illustrates how the filters are directed in the Fourier domain, and the lower part illustrates how the filter outputs are combined to produce an output vector.

Figure 2.10: Local representation on the arc of a circle: Orientation (middle) and principal direction/arc tangent (left).
It is first of all sensitive to the total energy of the neighbourhood. Two identical
neighbourhoods around \( f_1(\xi_{1a}, \xi_{2a}) \) and \( f_1(\xi_{1b}, \xi_{2b}) \), with the sole difference being the Fourier domain energy, will remain different in magnitude in the estimates
\( f_2(\xi_{1a}, \xi_{2a}) \) and \( f_2(\xi_{1b}, \xi_{2b}) \), while the arguments, of course, are identical.

In turn, the energy depends on the length of the vectors in the neighbourhood
of \( f_1(\xi_1, \xi_2) \) and long vectors there mean very certain estimates. The magnitude resulting from eq. (2.14) will thus in some sense preserve the certainty information
of the orientation estimates.

It goes without saying that vectors of high magnitude in the neighbourhood are
not enough to produce a high magnitude in \( h(\xi_1, \xi_2) \). The neighbourhood must
also be characterized by curvature in one direction, according to the assumption
made in eq. (2.10). This means that the magnitude also gives information of how
certain the curvature estimate is. Last but not least, the amount of curvature is also
embedded in the magnitude by the fit between the centre frequency of the filter and
the neighbourhood curvature \((2\pi \sqrt{a^2 + b^2} \text{ in eq. (2.10)})\). (The algorithms presented
in sections 2.2.2 and 2.2.3 describe the amount of curvature more explicit.)

The magnitude of \( f_2(\xi_1, \xi_2) \) contains, according to the discussion above, three
types of information:

- The certainty of the input data, i.e., how well the orientation estimates in the
  neighbourhood are defined.
- The fit to the curvature model, i.e., how well the assumption of one major
  curvature direction describes the neighbourhood.
- The curvature magnitude \( \kappa \) is implicitly reflected through the frequency char-
  acteristics of the filters used.

Bearing these three items in mind, it does not seem too extensive to call the mag-
nitude \( r \) of \( f_2(\xi_1, \xi_2) \) a certainty estimate.

Filter Shape

So far nothing specific has been said about the filters used to produce \( q_i \) in eq. (2.14).
A natural requirement is rotation invariance (also sometimes called magnitude
invariance). The magnitude of the curvature estimate should be independent of the
direction (tangent) of the curve, i.e., a curvature \( \kappa \) should result in an algorithm
output with the magnitude independent of where the curvature takes place. This
criterion can be used to discuss the choice of filters.

It would be advantageous if the quadrature filters (eq. (2.1)-(2.7)) used in the
orientation algorithm also could be used in the case of curvature estimation. Their
angular weighting function in the frequency domain is

\[
W_{\phi_k}(\phi) = \begin{cases} 
\cos^{2A}(\phi - \phi_k) & \text{if } \cos(\phi - \phi_k) \geq 0 \\
0 & \text{otherwise}
\end{cases}
\]  

where \( A \) is a parameter determining the angular shape of the filter. Normally four
complex filters, separated with \( \frac{\pi}{4} \) radians, are used for scalar orientation estimation.
For complex-valued images, four additional filters are used to cover the other half-
plane. These filters fulfill the criterion of magnitude invariance when used in the
Figure 2.11: Top: quadrature filter. Bottom: quasi-quadrature filter

orientation algorithm. Unfortunately they do not fulfil it for the curvature algorithm. The error is, however, very small (≈ 1%). It may very well be the case that the invariance error can be disregarded when the algorithm is used in an application. Otherwise another type of angular weight function can be used.

This function fulfils the magnitude invariance criterion (The proof is given in appendix 2.A). Filters with the angular weighting function (2.16) share many of the attributes of quadrature filters, e.g. the Hermitian property, and the name quasi-quadrature filters seems well motivated.

Fig. 2.11 illustrates a quadrature and a quasi-quadrature filter in the Fourier domain with identical radial function and the parameter $A$ of eq. (2.15) and (2.16) chosen as 1. Observe that the quasi-quadrature filter for larger values of $A$ approaches zero in one half-plane of the Fourier domain. This is demonstrated in fig. 2.12 where a quasi-quadrature filter with $A = 5$ is on display. The spatial response of a quasi-quadrature filter with $A = 1$ is visualized in fig. 2.13. This choice of $A$ yields an isotropic real part of the filter.

Variations of eq. (2.14) are of course possible. One such is to produce the vector

$$W_{\phi_k}(\phi) = \cos^{2A} \frac{\phi - \phi_k}{2}$$ (2.16)
Figure 2.12: A quasi-quadrature filter with $A = 5$

Figure 2.13: Spatial response of a quasi-quadrature filter with $A = 1$. Top: Real part. Bottom: Imaginary part
sum with the magnitudes of filter output raised to a power other than 1.

\[ f_2^B(\xi_1, \xi_2) = \sum_{k=1}^{K} q_k(\xi_1, \xi_2)^B e^{i\phi_k} \]  
(2.17)

\[ f_2 = \text{mag}(f_2^B)^{\frac{1}{2}} e^{i \arg f_2^B} \]  
(2.18)

The property of magnitude invariance must be tested for each case. For quasi-quadrature filters (eq. (2.16)) the following relation between \( A, B \) and \( K \) holds:

\[ K > AB + 1 \]  
(2.19)

This is proved for \( B \) being a positive integer in appendix 2.A. However, \( B \) need not necessarily be chosen as an integer. In fact, the use of quadrature filters becomes magnitude invariant for \( K = 8 \) and \( A = 1 \) combined with \( B = 1/2, 3/2 \) or \( 5/2 \) (appendix 2.B).

A discussion of good choices of \( A, B, \) and filter type (quadrature or quasi-quadrature) is given in chapter 2.3.1, where the different schemes are compared to each other with respect to discrete implementation and noise suppression.

**Complex Conjugate Filters**

For an even number of filters complex conjugate filters will oppose each other in the vector summation producing the curvature direction estimate (2.14). This can be used to decrease the computational complexity of the algorithm by deriving two filtering results 'for the price of one'. This is readily seen by comparison of the magnitude of the filter outputs \( q_k \) and \( q_k^* \) for two complex conjugate filters. The filters can be expressed as

\[ h_e(\xi_1, \xi_2) + ih_o(\xi_1, \xi_2) \]  
(2.20)

and

\[ h_e^*(\xi_1, \xi_2) + ih_o^*(\xi_1, \xi_2) = h_e(\xi_1, \xi_2) - ih_o(\xi_1, \xi_2) \]  
(2.21)

respectively. If the image \( f_1(\xi_1, \xi_2) \) is denoted as

\[ \text{re}(f_1(\xi_1, \xi_2)) + i\text{im}(f_1(\xi_1, \xi_2)) \]  
(2.22)

then the magnitude of the filter outputs can be expressed as (excluding the neighbourhood coordinates \((\xi_1, \xi_2)\))

\[ q_k = \sqrt{(h_e \ast \text{re}(f_1) - h_o \ast \text{im}(f_1))^2 + (h_e \ast \text{im}(f_1) + h_o \ast \text{re}(f_1))^2} \]  
(2.23)

and

\[ q_k^* = \sqrt{(h_e \ast \text{re}(f_1) + h_o \ast \text{im}(f_1))^2 + (h_e \ast \text{im}(f_1) - h_o \ast \text{re}(f_1))^2} \]  
(2.24)

where \( \ast \) denotes convolution or, when used as superscript, complex conjugate. This implies that two filter outputs can be obtained by one complex convolution if the convolution is implemented in cartesian complex form.
Furthermore, if the parameter \( A \) (see eq. (2.16) and fig. 2.13) is chosen as 1, the filter frequency function becomes

\[
H(u_1, u_2) = \frac{W_p(\rho) \cos(\phi - \phi_k)}{\text{even part}} + \frac{W_p(\rho)}{\text{odd part}}
\]  

(2.25)

This implies that all filters will have identical real parts, since \( H_e \) is independent of the filter direction. It is thus sufficient to convolve with one even and \( K/2 \) odd filters, e.g. to use four complex filters (8 scalar filters), separated by \( \pi/2 \) radians, only 3 scalar filters are needed.

The Curvature/Orientation Relation

The tangent and gradient of a curve are always perpendicular. This relation, \( \arg(f_1(\xi_1, \xi_2)) = 2\arg(f_2(\xi_1, \xi_2)) + \pi \), can be utilized in three different ways.

- **Consistency Estimate** Compute both curvature and orientation, and examine how well the estimates agree with each other.

- **Orientation Context** Compute the orientation value (or use the value in the orientation input image) and use that value to direct the curvature estimation, i.e., from the filters actually used, two filters are interpolated in the two possible curvature directions.

- **Curvature Context** Compute the curvature direction (possibly by orientation context!) and use that value to interpolate two filters for computation of the curved/straight measure. This is discussed in section 2.2.2.

The relation between curvature and orientation can thus be used to introduce a new parameter or to improve the curvature direction estimation as well as the curved/straight measure.

The relationship between gradient and tangent gives

\[
\begin{align*}
\arg(f_1) &= 2\phi = 2\theta + \pi \\
\arg(f_2) &= \theta_n = \frac{\arg(f_1) - \pi}{2} + n\pi, \quad n \in \{0, 1\}
\end{align*}
\]

(2.26) (2.27)

Non-fulfilment of this relationship indicates another type of neighbourhood, not describable as a single curve type. An example of such neighbourhoods are line ends \( \arg(f_1) = 2\arg(f_2) \). The use of relation (2.27) as a consistency estimate is thus a way to support evidence for the classification of neighbourhood type.

Another use of eq. (2.27) is as orientation context when the objective is to estimate the curvature direction. The orientation \( \phi \) of a neighbourhood is assumed to be known (e.g. by estimation). The orientation value constrains the curvature to take place in one of the two (opposite-directed) directions \( \theta_0 \) or \( \theta_1 \), and it is thus sufficient to interpolate two new filters in direction \( \theta_0 \) and \( \theta_1 \), and compute the difference between the outputs of these two filters, instead of computing \( \theta = \arg(\sum q_k e^{i\phi_k}) \).

The interpolation procedure has been discussed by Knutsson et al [63] and depends (of course) upon the shape of the filters used. An example of the interpolation procedure is given in appendix 2.C.
The discussion above implies that filtering with band-pass filters multiplied with $1, \cos \varphi, \sin \varphi, \cos 2\varphi, \ldots$ can be used as an alternative way to implement the filtering with the quasi-quadrature filter set in eq. (2.14). The alternative with direct use of the angular functions has advantages in convolution complexity ($\frac{5}{8}$ for the case of $\cos^4$). The algorithm contains other parts than convolutions, and the overall performance does of course depend on the hardware configuration. It can be advantageous on machines with little or no convolution support to implement eq. (2.18) as convolution with the angular functions followed by interpolation of $K$ quasi-quadrature filters.

Note that if the parameter $A$ is chosen to be a sufficiently high number (e.g., 6), then the achieved filter will be ‘almost’ zero on the opposite half-plane of the Fourier domain (fig. 2.12). Such filters can consequently be used for orientation estimation as well.

### 2.2.2 Curved/Straight Ratio

It is clear that the vector summation in eq. (2.14) is insensitive to the feature ‘orientation’, since orientation appears in the Fourier domain as ‘double spots’, i.e., the specific orientation energy is distributed on opposite sides of the origin (fig. 2.14). On the other hand, curvature can be seen as a ‘one spot’ event with most of the energy on one side of the origin (fig. 2.15). (Consider the extreme case with a neighbourhood of constant magnitude and the angular variation described by the one-dimensionality postulate formulated in eq. (2.10) which results in a one-peak Fourier transform, located at $u = a, v = b$.)

The computation of the curvature direction estimate via eq. (2.14) cancels out the orientation energy and detects curvature energy.
The fact that curvature and orientation energy are in a state of opposition can be used to compute a curved/straightness ratio. The orientation algorithm for scalar images was discussed earlier, where the Hermitian property was utilized, as well as its generalization to vector orientation which does not perform any check that both sides of a line through the origin of the frequency domain have equal strength. The curvature direction estimation method has also been described, a method which is built on the fact that curvature is reflected as unequal strength in the Fourier domain with respect to the origin. It is also possible to do the opposite, i.e., detect equal strength.

'Straightness'

The complementary neighbourhood-event of curvature is neighbourhoods where the curve is a straight line, i.e., all orientation vectors in the neighbourhood have the same argument. This 'non-curvature' event can be referred to as straightness and can be detected by first subtracting the energy detected as 'curvature energy' and then applying the vector orientation algorithm on the remaining energy. It follows from the discussion of relations between complex conjugate filters at the beginning of section 2.2.1 that, for a specific complex conjugate filter pair \((k, K/2 + k)\), the straight line component can be expressed as

\[
l_k = q_k + q_{k/2+k} - |q_k - q_{k/2+k}|
\]  

(2.28)

The straight line energy contribution from the different filter pairs can then be combined to form an estimate of straight line orientation, i.e.

\[
f'_2(\xi_1, \xi_2) = \sum_{k=1}^{K} l_k(\xi_1, \xi_2)e^{2\pi i k}
\]  

(2.29)
The curved/straight measure can then be computed in two ways. The first and maybe the most natural variant is

$$\zeta = \text{mag}(f_2) - \text{mag}(f'_2)$$

(2.30)

The drawback with this approach is that the \(\text{mag}(f'_2)\) will tend to be much larger than \(\text{mag}(f_2)\). A remedy is to compute the straightness in another direction than the dominant using the gradient/tangent relation discussed in section 2.2.1 in the following way: interpolate two filters in the candidate directions for curvature and compute the curved/straightness measure for that filter pair.

It is advantageous to add a parameter \(C\) in the comparison of the two vector sums and normalize the range of \(\zeta\) to \(\pm 1\). This gives a new version of eq. (2.30).

$$\zeta = \frac{C\text{mag}(f_2) - \text{mag}(f'_2)}{C\text{mag}(f_2) + \text{mag}(f'_2)}$$

(2.31)

Moreover, it is similarly advantageous if the scalar measure \(\zeta\) can be represented as a vector, where information is contained in vector arguments and certainty estimates are used as vector magnitudes, thus making it possible to use all algorithms on all algorithm results.

$$f''(\xi_1, \xi_2) = (\text{mag}(f_2) + \text{mag}(f'_2)) \exp(j(\zeta \cdot \frac{\pi}{2} + \frac{\pi}{2}))$$

(2.32)

This constitutes a curved/straight phase image with argument values in the range \([0 \ldots \pi]\). The curved/straight measure can now be denoted 'ratio', since the difference is now contained in the exponent.

This algorithm should be implemented with quasi-quadrature filters with \(A \geq 2\) (see the proof given in appendix 2.A).

The curved/straight measure is one way of estimating \(\kappa\), although it is not easy to map the estimate to an exact value. Later in this chapter it is described how the estimation of \(\kappa\) can be done in a way which can be mapped more easily.

### 2.2.3 Curvature Magnitude

The curvature \(\kappa\) corresponds to the distance between the origin and the centre of gravity in the local Fourier spectrum of the orientation image. This distance can be estimated by modification of algorithms estimating the local frequency. A local frequency estimation method was described by Knutsson in [64]. The frequency estimate is obtained by a combination of the magnitudes from two or more filter sets with certain relations in the filter parameters. The one-dimensionality assumption in this case corresponds to the presence of one single frequency in the neighbourhood, and the magnitude, as before, plays the role of certainty estimate, i.e. indicates how dominant the estimated frequency is. Other algorithms on the same theme are for instance presented in Näppä and Granlund [76] and Haglund et al [44]. Implementation details and overall design differ considerably but there is a common scheme shared by the algorithms:

1. Filter the input data with a number of different filter sets, where each set has a different centre frequency, e.g. low, medium and high for three filter sets.
2. Combine the filter outputs from each filter set to form an estimate of the frequency domain energy for the centre frequency and bandwidth of the filter set.

3. Combine the energy estimates from the different filter sets to form an estimate of the frequency.

The same scheme can with some changes be used to estimate the amount of curvature \( \kappa \). Recall from the earlier discussion regarding the information content in the magnitude of the vector obtained by eq. (2.14) that the magnitude of the curvature direction vector \( f_2 \) depends upon three different things: the certainty of the orientation data; the fit to the curvature model; and the relationship between the curvature magnitude \( \kappa \) and the frequency characteristics of the filters used. The two first items depend solely on the neighbourhood. Only the last item on the list is subject to change if different filter sets are used in the same neighbourhood. This implies that the following procedure can be used for estimation of \( \kappa \) where differences to the original frequency estimation scheme are emphasized with italics:

1. Filter the orientation data with a number of quasi-quadrature filter sets, where each set has a different centre frequency.

2. Combine the filter outputs from each filter set using eq. (2.14) to form an estimate of the curvature for this particular centre frequency and bandwidth.

3. Combine the magnitudes of the curvature direction estimates from the different filter sets to form an estimate of the magnitude of the curvature \( \kappa \).

The certainty estimate can be further constrained. The original local frequency algorithm is applied to the grey level image and does not discern different kinds of events, e.g., ‘orientation’, ‘curvature’ or other types of neighbourhoods. The curvature ‘frequency’ algorithm is applied to the orientation vector image, and a certainty estimate originating from the curvature direction estimation can be included, together with the frequency estimation certainty, to form a total certainty of the estimate.

How should the curvature direction certainty estimate be computed? The magnitude of the vector obtained by eq. (2.14) is one version of such an estimate but is already used in the computation of the value of \( \kappa \). It is desirable to achieve a certainty value which does not depend upon the curvature magnitude \( \kappa \).

One formulation of certainty is the quotient between energy according to the model versus total energy. This can be expressed as

\[
r = \frac{V}{M} \tag{2.33}
\]

where \( V \) is the magnitude of the vector produced by vector addition, such as (2.5) or (2.14), and \( M \) is the scalar sum of the filter magnitudes. Special care must be taken when \( M \) approaches zero. The easiest way is to add a small term in the denominator to avoid division by zero.

The different certainty estimates from the curvature direction estimations and from the frequency estimation should be combined into one single certainty value.
This faces us with the problem of combining evidence. Not only is there the problem of combining two unrelated certainty estimates, but also the difficulty of combining the curvature statements from two or more different sets of filters.

Each filter set $k$ produces a certainty estimate $r_k$ with components $V_k$ and $M_k$, all according to the formulation in eq. (2.33). The frequency algorithm builds on the fact that the filter sets used have a certain overlap in frequency coverage. This means that the different $r_k$ are not independent. The following scheme is proposed, where account also is taken of the arguments of the vector sums.

$$2 \left| \sum k V_k e^{i \arg V_k} \right| \sum_k M_k$$

The final certainty measure is then achieved by multiplication of the two (unrelated) certainty estimates obtained by eq. (2.34) and the frequency algorithm.

The performance of the curvature magnitude algorithm is illustrated later in this thesis. The certainty measure is but one of the components in the algorithm, and no examples which isolate and evaluate the performance of the certainty estimation are given here, but the good qualities of the certainty estimate are exemplified on synthetic data in [9].

### 2.3 Results

The curvature algorithms described in chapter 2.2 have been exhaustively tested. This section gives examples and summarizes the results. Examples are given of processing on both synthetic and real images. The former ones give the necessary control of the input environment to indicate the performance limits of the algorithms, and the latter one demonstrates that natural scenes by all means can be handled with respect to curvature.

Visualization of complex-valued images can be made in a number of ways, as illustrated in fig. 2.16. A combination of colour for representation of the argument information and brightness for the magnitude gives natural and intuitive image interpretation. Magnitude and argument can be viewed separately when colour is not available, although this is not equally easy to comprehend. The third alternative is, as in fig. 1.3, to actually draw the vector field.

#### 2.3.1 Synthetic Vector Images

Synthetic vector images provide valuable ground proof and enable an isolated study of the curvature algorithms performance without taking the orientation algorithm into account. An extensive examination of the sensitivity of the curvature direction algorithm has been carried out and the results are summarized here. The performance of the other algorithms has also been verified. A detailed documentation of the extensive noise analysis with results and discussions of the conclusions made can be found in chapter 3.

The choice of $B$. The summation exponent $B$ in eq. (2.18) is best chosen as 1. This choice ensures that the magnitude ($\approx$ certainty) decreases when the noise
The accuracy of the direction estimate is, however, improved with \( B = 2 \), and the best performance is achieved with a mixed algorithm where the magnitude is computed with \( B = 1 \) and the argument is determined with \( B = 2 \).

**The choice of angular function: Quadrature or quasi-quadrature filters**

Narrow quasi-quadrature filters (e.g., \( \cos^{12} \frac{x - \phi}{2} \)) resemble quadrature filters to a wide extent. Quadrature filters have a small disadvantage in the lack of magnitude invariance of direction for \( B = 1 \), and \( B \neq 1 \) gives clear disadvantages with respect to magnitude. Thus quasi-quadrature filters seem somewhat better suited for the purpose of curvature estimation.

**How to choose \( K \), the number of filters**

The performance is improved for an increase in the number of filters used, although the slope of the performance curve decreases quickly. The choice is of course a balance between accuracy and computational complexity, but the shape of the curve indicates that the choice will never be far from the minimum number of filters, three filters in the case of \( \cos^2 \), four in the case of \( \cos^4 \), etc.

**How to choose \( A \), the angular shape of the filter**

Changes in the parameter \( A \) of eq. (2.16) have larger effect in the performance than changes in \( K \), the number of filters used. Narrow filters, i.e., high choices of \( A \), at first seem to perform best. On the other hand does a high value on \( A \) imply more filters, and thus increases computational complexity. However, the situation changes dramatically when vector averaging is added as post processing. A small average filter decreases the difference between broad and narrow filters, and a large average filter gives an advantage to broad filters! The choice of \( A \) should be directed by the choice of the vector average filter, a choice which depends on the spatial resolution restriction for the specific image processing task. This is discussed more in detail in chapter 3.

**Use of orientation context**

The use of orientation context (eq. (2.27)) to restrict the curvature to take place in the directions indicated by the orientation estimate

\[ \text{Figure 2.16: Colour can be used to visualize a complex-valued pixel (left). When colour is not at hand one can display the magnitude (middle) and the argument (right) separately.} \]
2.3.2 Synthetic Grey Level Images

An image containing a sine wave pattern is with respect to curvature an adequate test, since the curve contains points with curvature values in a range from zero to high magnitude. Such an image is displayed in the upper part of fig. A.1 in appendix A, where the intensity of the sine pattern decreases to the right. White noise has been added to the image to make the curvature estimation more difficult.

The middle part of fig. A.1 shows in a thresholded fashion the output from the curvature direction algorithm. The lower part displays the curved/straight ratio with green ($0^\circ$) representing linear structures and blue ($\pi/2$) representing equal shares of curvature and linearity. The magnitude represents the certainty measure.

Another natural test pattern for curvature algorithms are images containing circles of different radii, since

$$\kappa = \frac{1}{r}$$

where $r$ denotes the radius of the circle. Fig. 2.17 is constituted by ten different circles with a space of half an octave between the different radii. The same procedure as in fig. A.1 has been used to make the discrimination between different amounts of curvature more difficult. That is, the pattern intensity decreases to the right, and white noise has been added. This image is used for the comparison between one of the algorithms presented here, the curved/straight ratio, and the straightforward differential geometry approach.

Figure 2.17: Ten different sizes of circles together with white noise increases the accuracy.
The straight-forward algorithm works in our implementation quite well on noise-free images. The first and second derivatives of eq. (1.2) are estimated by filters (see appendix 2.6 for the details on filter design) and a (small) constant has been added to the denominator in the implementation to avoid problems when the gradient approaches zero.

The lack of certainty mechanism makes, however, the algorithm fail on more complex images as indicated in the result in fig. 2.18. The hierarchical feature pyramid with a sequence of orientation estimation, vector averaging and the curved/straight ratio gives a better result as indicated in fig. A.2 and A.3.

The accuracy of the curvature magnitude algorithm is demonstrated in fig. 2.19 where the argument (the value) is plotted on the y-axis for the ten circles in fig. 2.17, ordered on the x-axis. This experiment was carried out on an image without noise.

2.3.3 Natural Images

The final example is the well-known ‘Lenna’ image from the image database of the USC Image Processing Institute (fig. 2.20).

The first step is to attain an orientation description, displayed in fig. A.4. The orientation image is then rescaled down to $256^2$. Applying the curvature direction algorithm on that image yields the result in fig. A.5. Note how well the irises of the eyes are detected. Other high curvature spots are for instance locks of the hair and abrupt orientation changes in linear structures in the background.

The result from using the differential geometry algorithm on a $256^2$ ‘Lenna’ is presented in fig. 2.21. In this case, the hair is detected as the area of most curvature.
Figure 2.19: A plot of the curvature magnitude output for the image with ten circles (but without noise!). Note the linear region between the third and seventh circle. The dotted line indicates the correct value of $\kappa$.
Figure 2.20: Lenna

Figure 2.21: The output from the 'standard' differential geometry algorithm
The eyes are outlined but the irises are not detected.

Fig. A.6 is an example of the output from the curvature magnitude algorithm. Here some extra steps have been taken. The arguments of the rescaled orientation image have been doubled to make a better fit between the image curvature and the frequency characteristics of the filters. (eq. (2.11) and fig. 2.7) The output has been vector averaged with a $7 \times 7$ filter to give areas uniform values. Yellow colour corresponds to low curvature, green to middle, blue to high and red indicates a mixture between low and high curvature.

Fig. A.7 examplifies the curved/straight ratio. Once again the arguments in the orientation image are doubled. Linear structures are marked with green, mixtures with blue and curvature is coloured with red.

## 2.4 Conclusions

Reliable curvature estimation requires non-thresholding methods, and existing algorithms in general have lacked that essential and progressive feature. The algorithms presented in this chapter use a hierarchical strategy, where the curvature is estimated on a gradient-equivalent image derived from the grey level information. Both steps are performed without thresholding and the base operation is convolution.

Emphasis has been placed on frequency domain features, and a model of how curvature is reflected in the Fourier domain has been proposed. A great deal of details concerning implementation have been given to facilitate for these methods to be implemented and used elsewhere. A very appealing fact is that the concept of curvature, and the description thereof, in the interpretation given here becomes very similar to the earlier applied methods for orientation description.

## Appendices

### 2.A Magnitude Invariance of Quasi-Quadrature Filters

This proof is adapted from a similar proof for quadrature filters in Knutsson [64]. Assume that the complex-valued neighbourhood has a local Fourier transform constituted by one impulse located outside the origin, where the coordinate vector of this impulse has the argument $\varphi_z$.

Apply $K$ quasi-quadrature filters (eq. (2.16)) to the neighbourhood. The magnitude of the filter output of filter $k$ is given by

$$X_k = X_a \cos^{2A} \frac{\varphi_z - \frac{2\pi k}{K}}{2}$$

(2.36)

where $X_a$ is the impulse response of the radial weight function of the quasi-quadrature filter.

The sum of $K$ vectors with magnitude given by eq. (2.36) and the angle of the $k$-th vector being $M \cdot \frac{2\pi k}{K}$. Where $M = 0, 1, 2, \ldots$ can be written

$$Z_a =$$

41
\[ \begin{align*} 
&= \sum_{k=0}^{K-1} \exp\left(i \frac{M2\pi k}{K}\right) \cdot X_k = \sum_{k=0}^{K-1} \exp\left(i \frac{M2\pi k}{K}\right) \cdot X_a \cos^2 \left( \frac{\varphi_a}{2} - \frac{\pi k}{K} \right) 
\end{align*} \] (2.37)

and hence
\[ \tilde{Z}_a = X_a 2^{-2A} \sum_{k=0}^{K-1} \exp\left(i \frac{M2\pi k}{K}\right) \left[ \exp\left(i \left(\frac{\varphi_a}{2} - \frac{k\pi}{K}\right)\right) + \exp\left(-i \left(\frac{\varphi_a}{2} - \frac{k\pi}{K}\right)\right) \right] 2^A \] (2.38)

The term in brackets can be written as
\[ \sum_{n=0}^{2A} \left( \begin{array}{c} 2A \\ n \end{array} \right) \exp\left(i \left(\frac{\varphi_a}{2} - \frac{n\pi}{K}\right)\right) \cdot \exp\left(-i (2A - n) \left(\frac{\varphi_a}{2} - \frac{k\pi}{K}\right)\right). \] (2.39)

Rewriting the bracket term, rearranging the summing order and moving the \(k\pi/K\) terms together yields
\[ \tilde{Z}_a = X_a 2^{-2A} \sum_{n=0}^{2A} \left( \begin{array}{c} 2A \\ n \end{array} \right) \exp\left(i \varphi_a (n - A)\right) \cdot \left[ \sum_{k=0}^{K-1} \exp\left(i \frac{k2\pi}{K} (M - A - n)\right) \right] \] (2.40)

then if \( M \leq A \)
\[ \tilde{Z}_a = X_a \cdot 2^{-2A} \cdot \left[ \left( \begin{array}{c} 2A \\ M + A \end{array} \right) \exp\left(iM\varphi_a\right) \cdot K + \sum_{n=0}^{2A} \left( \begin{array}{c} 2A \\ n \end{array} \right) \exp\left(i \varphi_a (n - A)\right) \cdot \frac{1 - \exp\left(i2\pi (M + A - n)\right)}{1 - \exp\left(i\frac{2\pi}{K} (M + A - n)\right)} \right] \] (2.41)

then for \( K > M + A \) we have
\[ \tilde{Z}_a = X_a \cdot K \cdot 2^{-2A} \left( \begin{array}{c} 2A \\ M + A \end{array} \right) \cdot \exp\left(iM\varphi_a\right) \] (2.42)

if \( M = 1 \) (eq. (2.14)) then
\[ \begin{align*} 
\arg(\tilde{Z}_a) &= \varphi_a \\
|\tilde{Z}_a| &= X_a \cdot K \cdot 2^{-2A} \cdot \frac{(2A)!}{(A+1)! (A-1)!} 
\end{align*} \] (2.43)

if \( M = 0 \) (scalar summation) then
\[ Z_{a0} = \sum_{k=0}^{K-1} X_k = X_a \cdot K \cdot 2^{-2A} \frac{(2A)!}{(A!)^2} \] (2.44)

The \( A \) in this proof corresponds to \( AB \) in eq. (2.19). Note that there are two relationships that should be fulfilled.
\[ \begin{align*} 
&= \begin{cases} 
M &\leq A \\
K &> M + A 
\end{cases} 
\end{align*} \] (2.45)

This implies that the 'curved/straight' algorithm described in chapter 2.2.2 should use quasi-quadrature filters with \( A \geq 2 \) since this algorithm uses both curvature direction and vector orientation (\( M = 2 \)).
2.B Using Quadrature Filters in Equation (2.18)

It was earlier stated that eq. (2.18) is magnitude invariant for quadrature filters and $K = 8$, $A = 1$ and $B = 1/2$, 3/2 or 5/2.

Only four of eight filters will give any contribution if the Fourier domain consists of a single line with orientation $\varphi$ ending in the origin. Let without loss of generality $\varphi \in [0, \frac{\pi}{4}]$ and $\phi_k = 0$. The vector sum for $B = \frac{1}{2}$ becomes

$$f_{\frac{1}{2}} = X_{\frac{1}{2}}^k [\cos \varphi + \frac{1+i}{\sqrt{2}} \cos(\varphi + \frac{\pi}{4}) + i \cos(\varphi + \frac{\pi}{2}) + \frac{1-i}{\sqrt{2}} \cos(\varphi + \frac{7\pi}{4})] =$$

$$= X_{\frac{1}{2}}^k [\cos \varphi + \frac{1}{\sqrt{2}} \cos \varphi + \sin \varphi + 0 + \frac{1}{\sqrt{2}} \cos \varphi - \sin \varphi +$$

$$+ i [0 + \frac{1}{\sqrt{2}} \cos \varphi + \sin \varphi + \sin \varphi + \frac{1}{\sqrt{2}} \cos \varphi - \sin \varphi)] =$$

$$= 2X_{\frac{1}{2}}^k [\cos \varphi + i \sin \varphi]$$  \hspace{1cm} (2.46)

The corresponding equation for $B = \frac{3}{2}$ is

$$f_{\frac{3}{2}} = X_{\frac{3}{2}}^k [\cos^3 \varphi + \frac{1+i}{\sqrt{2}} \cos^3(\varphi + \frac{\pi}{4}) + i \cos^3(\varphi + \frac{\pi}{2}) + \frac{1-i}{\sqrt{2}} \cos^3(\varphi + \frac{7\pi}{4})] =$$

$$= X_{\frac{3}{2}}^k [\cos^3 \varphi + \frac{1}{\sqrt{2}} \cos^3 \varphi + \sin^3 \varphi + 0 + \frac{1}{\sqrt{2}} \cos^3 \varphi - \sin^3 \varphi +$$

$$+ i [0 + \frac{1}{\sqrt{2}} \cos \varphi + \sin \varphi + \sin \varphi + \frac{1}{\sqrt{2}} \cos \varphi - \sin \varphi)] =$$

$$= X_{\frac{3}{2}}^k [\cos^3 \varphi + \frac{1}{2} \cos^3 \varphi + \frac{3}{2} \cos \varphi \sin^2 \varphi + i \left[ \frac{3}{2} \cos^2 \varphi \sin \varphi + \frac{1}{2} \sin^3 \varphi + \sin^3 \varphi \right]] =$$

$$= \frac{3}{2} X_{\frac{3}{2}}^k [\cos \varphi + i \sin \varphi]$$  \hspace{1cm} (2.47)

And for $B = \frac{5}{2}$

$$f_{\frac{5}{2}} = X_{\frac{5}{2}}^k [\cos^5 \varphi + \frac{1+i}{\sqrt{2}} \cos^5(\varphi + \frac{\pi}{4}) + i \cos^5(\varphi + \frac{\pi}{2}) + \frac{1-i}{\sqrt{2}} \cos^5(\varphi + \frac{7\pi}{4})] =$$

$$= X_{\frac{5}{2}}^k [\cos^5 \varphi + \frac{1}{\sqrt{2}} \cos^5 \varphi + \sin^5 \varphi + 0 + \frac{1}{\sqrt{2}} \cos^5 \varphi - \sin^5 \varphi +$$

$$+ i [0 + \frac{1}{\sqrt{2}} \cos \varphi + \sin \varphi + \sin \varphi + \frac{1}{\sqrt{2}} \cos \varphi - \sin \varphi]] =$$

$$= X_{\frac{5}{2}}^k [\cos^5 \varphi + \frac{1}{2} \cos^5 \varphi + \frac{5}{2} \cos^3 \varphi \sin^2 \varphi + \frac{5}{4} \cos^2 \sin^4 \varphi +$$

$$+ i \left[ \sin^5 \varphi + \frac{1}{4} \sin^5 \varphi + \frac{5}{2} \sin^3 \varphi \cos^2 \varphi + \frac{5}{4} \sin \varphi \cos^4 \varphi \right]] =$$

$$= \frac{5}{4} X_{\frac{5}{2}}^k [\cos \varphi + i \sin \varphi]$$  \hspace{1cm} (2.48)

2.C The Filter Interpolation Procedure

Assume for instance that the quasi-quadrature filters are defined with $K = 8$ and $A = 2$.

One wants to interpolate the filters:

$$H(\varphi) = \cos^4 \varphi - \theta_n$$  \hspace{1cm} (2.49)
Since
\[
\cos^4 v = \frac{1}{8}(3 + 4 \cos 2v + \cos 4v)
\]

\(H(\varphi)\) can be rewritten to
\[
H(\varphi) = \frac{1}{8}(3 + 4 [\cos \varphi \cos \theta_n + \sin \varphi \sin \theta_n] + \cos 2\varphi \cos 2\theta_n + \sin 2\varphi \sin 2\theta_n)
\]

(2.51)

This requires the computation (i.e., filtering) of the following amounts: 1, \(\cos \varphi\), \(\sin \varphi\), \(\cos 2\varphi\) and \(\sin 2\varphi\). One can either use filters with such angular functions or use the existing quasi-quadrature filter set. With \(\phi_k = \frac{(k-1)\pi}{4}, k \in \{1, 2, \ldots, 8\}\)

\[
H_1(\varphi) = \frac{1}{8}(3 + 4 \cos \varphi + \cos 2\varphi)
\]
\[
H_2(\varphi) = \frac{1}{8}(3 + \frac{4}{\sqrt{2}}(\cos \varphi + \sin \varphi) + \sin 2\varphi)
\]
\[
H_3(\varphi) = \frac{1}{8}(3 + 4 \sin \varphi - \cos 2\varphi)
\]
\[
H_4(\varphi) = \frac{1}{8}(3 + \frac{4}{\sqrt{2}}(\sin \varphi - \cos \varphi) - \sin 2\varphi)
\]
\[
H_5(\varphi) = \frac{1}{8}(3 - 4 \cos \varphi + \cos 2\varphi)
\]
\[
H_6(\varphi) = \frac{1}{8}(3 + \frac{4}{\sqrt{2}}(-\cos \varphi - \sin \varphi) + \sin 2\varphi)
\]
\[
H_7(\varphi) = \frac{1}{8}(3 - 4 \sin \varphi - \cos 2\varphi)
\]
\[
H_8(\varphi) = \frac{1}{8}(3 + \frac{4}{\sqrt{2}}(\cos \varphi - \sin \varphi) - \sin 2\varphi)
\]

(2.52) - (2.59)

The filters can be combined in the following way to express the wanted quantities:
\[
\sum H_k = 3
\]
\[
H_1 + H_5 = \cos \varphi
\]
\[
H_3 - H_7 = \sin \varphi
\]
\[
H_1 + H_5 + H_3 - H_7 = \frac{1}{2} \cos 2\varphi
\]
\[
H_2 + H_4 + H_6 - H_8 = \frac{1}{2} \sin 2\varphi
\]

(2.60) - (2.64)

2. D Design of Filters for the ‘Standard’ Algorithm (eq. (1.2))

The filters are designed in the Fourier domain with use of three building blocks:
\[
X(u_1, u_2) = u_1
\]
\[
Y(u_1, u_2) = u_2
\]
\[
G(u_1, u_2) = \exp\left(-0.49\rho^2\right)
\]

(2.65) - (2.67)

where \(\rho = \sqrt{u_1^2 + u_2^2}\) expressed in pixel units.

The differentiation filters are achieved by an inverse DFT computed over a 16 x 16 neighbourhood and disposal of the outer row and column, keeping 15 x 15 points.
\[
\frac{\partial f}{\partial x} = F^{-1}\{G(u_1, u_2) \cdot X(u_1, u_2)\} \quad (2.68)
\]
\[
\frac{\partial f}{\partial y} = F^{-1}\{G(u_1, u_2) \cdot Y(u_1, u_2)\} \quad (2.69)
\]
\[
\frac{\partial^2 f}{\partial x^2} = F^{-1}\{G(u_1, u_2) \cdot X(u_1, u_2) \cdot X(u_1, u_2)\} \quad (2.70)
\]
\[
\frac{\partial^2 f}{\partial x \partial y} = F^{-1}\{G(u_1, u_2) \cdot X(u_1, u_2) \cdot Y(u_1, u_2)\} \quad (2.71)
\]
\[
\frac{\partial^2 f}{\partial y^2} = F^{-1}\{G(u_1, u_2) \cdot Y(u_1, u_2) \cdot Y(u_1, u_2)\} \quad (2.72)
\]

where \( \cdot \) denotes pointwise multiplication.
Chapter 3

A Filtering Strategy for Orientation and Curvature Description

This chapter discusses the hierarchical combination of the orientation and curvature estimation algorithms and how to get the ‘optimal’ performance from the algorithms, e.g., what type of and how many filters to use, should subsampling be used prior to the curvature estimation etc. Some of these issues were summarized for the curvature algorithm in chapter 2.3.1, where the 2D curvature algorithms were presented.

The algorithms presented in chapter 2 measure different features by convolving the image with different filter functions. For such algorithms there is a question of priorities between specificity and computational cost. This chapter shows that the use of vector averaging as a post-processing procedure eliminates the need to have many specialized filters. In fact, the performance of few broad band filters followed by a small averaging filter is better than the use of many narrow band filters! This methodology is applicable for both the orientation and the curvature direction estimation.

3.1 Background

The word ‘optimal’ was used in the description above. Numerous image processing algorithms for feature extraction have been developed and many of them extract the same type of features, e.g., edges. It is, however, not always clear which type of method is preferable. This is not only a case of a trade-off between specificity and computational cost. It is also a question of how to measure the performance between the different algorithms.

Different types of comparison have been made, e.g., Pratt [83, ch. 17.5], and Danielsson and Seger [31], both dealing with edge operators. The result of a comparison depends upon how the examination is done with factors such as the assumptions made, test images, performance measures etc.

This chapter evaluates the extraction of orientation and curvature features with respect to spatial resolution and noise suppression. The algorithms are applied with different numbers of filters, different filter functions and different normalization on synthetic images. The effect of averaging as post-processing is also examined.
3.2 Normalization

Different ways of computing the curvature direction estimate were presented in chapter 2. One more type of normalization, also applicable to the orientation estimation (Knutsson [66]), is now introduced.

Recall from the proof in chapter 2.4 that the vector sum producing the curvature direction estimate (eq. (2.43)) and the scalar sum of the filter magnitudes (eq. (2.44)) have different magnitudes. The following relationship holds for neighbourhoods fulfilling the curvature model,

\[
\frac{\| \sum q_k \exp(i \phi_k) \|}{\sum q_k} = \frac{A!}{(A+1)! \cdot (A-1)!}
\]

where \( A \) is the parameter in eq. (2.16). This equation holds true for orientation estimation and quadrature filters (eq. (2.15)) as well.

This makes it possible to check how consistent the neighbourhood is with respect to the model. The magnitude of the estimate should be reduced whenever the relation in eq. (3.1) does not hold. This is especially important in neighbourhoods where the curvature direction estimate (eq. (2.14)) produces vectors with magnitudes larger than predicted by eq. (3.1). An example would be a neighbourhood in which two events interfere in such a way to influence the filter outputs.

One way of reducing the magnitude is by introducing a function similar to the certainty normalization introduced in eq. (2.33), i.e., a quotient between energy according to the model versus total energy. The energy according to the model is computed as the length of the vector produced by the vector summation in eq. (2.14).

\[
V = \| \sum q_k \exp(i \phi_k) \|
\]

The total energy is estimated by taking the scalar sum of the filter outputs and rescale it with eq. (3.1).

\[
M = \frac{A!}{(A+1)! \cdot (A-1)!} \sum q_k
\]

Finally, combine the two magnitudes and introduce the combination as a consistency measure into eq. (2.14).

\[
\frac{2VM}{V^2 + M^2} \sum q_k \exp(i \phi_k)
\]

This consistency function is plotted in fig. 3.1 with \( V/\sum q_k \) on the horizontal axis. The angular bandwidth is in this case \( A = 1 \). The same consistency measure strategy can, as stated above, also be used in the orientation estimation. The improvement obtained with this modification is discussed in chapter 3.4 (e.g., fig. 3.6).

3.3 Performance Measures

The representation of the local orientation and the local curvature both consist of a magnitude, describing the certainty of the estimated value, and an argument, representing the value. The magnitude and the argument are for that reason examined
both separately and together. The evaluation is done on synthetic images where the estimated values are compared to the theoretical values with a number of performance measures. Definitions of the measures and some of the test images can be found in appendix 3.A. The performance measures can be described in words as follows:

**Magnitude Invariance** Although the algorithm may ensure magnitude invariance (introduced in the discussion resulting in eq. (2.19)), the implementation with quantization errors etc. may be cumbersome.

**Average Magnitude** It should decrease when the noise increases. It is on the other hand a good quality to keep the magnitude to a certain extent, since it reflects the amount of information extracted by the algorithm.

**Bias** No specific value should be preferred to another.

**Angular Error** The difference between actual and estimated value. No regard is taken to the magnitude, i.e. the certainty of the estimate.

**Weighted Angular Error** Does also take the magnitude into account.

### 3.4 Results

The performance measures were computed on the results achieved by processing test images with different amounts of additive gaussian noise. The test pattern for orientation estimation is shown in fig. 3.2. A similar complex-valued pattern was used for the curvature direction algorithm (see appendix 3.A). Two of the measures, **Magnitude Invariance** and **Bias**, were of no use since all the candidate filter functions
Figure 3.2: The orientation test pattern

Figure 3.3: The effect of an increased number of filters. Note the scaling on the y-axis!
fulfilled these criteria. The results of the computation of the three other performance measures lead to the following conclusions.

The performance is improved for an increase in the number of filters used, $K$, although the slope of the performance curve decreases quickly as indicated in fig. 3.3. The choice is of course a balance between accuracy and computational complexity, but the shape of the curve indicates that the choice will never be far from the minimum number of filters.

Changes of the angular bandwidth, $A$, have larger effect in the performance than changes in $K$. Narrow filters, i.e. high choices of $A$, at first seem to perform best. On the other hand does a high value on $A$ imply more filters, i.e. increase the computational complexity, since the relation (see chapter 2.A)

$$K > A + 1$$

must be fulfilled to ensure magnitude invariance.

The addition of vector averaging as post processing induces some changes. The conclusion made for $K$ is still valid but the situation for $A$ is changed. A small average filter decreases the difference between broad and narrow filters, and a large average filter gives an advantage to broad filters! A possible explanation to this behaviour is that a narrow banded filter will, since it is concentrated in the Fourier domain, be wide in the spatial domain and thus have a sort of 'built-in averaging'.

The vector averaging means a loss of spatial resolution, but that does not matter on patterns like the one in fig. 3.2. Another type of test image is used to examine how the maximum amount of detail is extracted. Fig. 3.4 shows a detail of the test pattern used to evaluate the orientation algorithm. A similar complex-valued
pattern was used for the curvature direction algorithm. The patterns are both partitioned into squares, so that all types of abrupt orientation (or curvature) changes take place in the image. Fig. 3.5 shows the result; a broad band filter followed by a small vector average filter is better than a narrow band filter.

The normalization introduced in chapter 3.2 decreases the magnitude in noisy neighbourhoods and consequently gives better performance. The weighted angular error decreases with up to 10%. This is indicated in fig. 3.6, where the average magnitude and the weighted angular error are plotted before and after normalization.

3.5 Summary

The performance of the orientation algorithm and the curvature direction algorithm has been examined. The rather unexpected conclusion is that a few filters followed by averaging with a small filter gives the best performance. This have nice implications on the computational complexity. The parameter A should be chosen to 1 in the filter design (eq. (2.15) and eq. (2.16)) and the normalization procedure described in chapter 3.2 should be used. The choice of vector average filter depends on the spatial resolution restriction for the specific image processing task.

The filtering strategy is summarized in fig 3.7. Four complex filters each for the orientation estimation and the curvature direction estimation give easy implementations with relatively low computational complexity. The grey ellipse symbolizes resampling and/or multiplication of the argument of the orientation estimates.
Figure 3.6: Average magnitude and weighted angular error before (solid) and after (dashed) normalization with eq. (3.4). Note the scaling of the vertical axis!
Figure 3.7: The filtering strategy. The grey ellipse represents optional resampling of the image and/or rescaling of the vector arguments.
Appendices

3.A Test Patterns and Error Measures

The curvature direction test pattern corresponding to fig. 3.2 Let \( r \) be the normalized distance from the centre of the image such that \( r \in [0, \sqrt{2}] \). The test pattern \( \rho e^{j\varphi} \) is then defined as

\[
\rho(r) = \begin{cases} 
0 & r \leq \frac{\sqrt{2}}{2} \\
\cos^2(16\pi r - 4\pi) & \frac{\sqrt{2}}{2} \leq r \leq \frac{1}{2} \\
1 & \frac{1}{2} \leq r \leq \frac{3}{4} \\
\sin^2(4\pi(r - 1)) & \frac{3}{4} \leq r \leq 1 \\
0 & 1 \leq r 
\end{cases} \tag{3.6}
\]

and

\[
\varphi(r) = \frac{112\pi}{\ln 2} (2^{-\frac{90.5}{10}} - 2^{-\frac{90.5}{2}r}) \tag{3.7}
\]

The expressions (3.6) and (3.7) are suitable for the generation of a 512 x 512 test pattern. The correct result on this test pattern is an image with vectors directed as \( \xi'_1 + i\xi'_2 \) for spatial coordinates \( \xi' \) with the origin \((0, 0)\) in the centre of the image.

The following expressions were used to get images with comparable signal to noise ratio. The signal energy was held constant, while the noise was increased in the different images. Observe that the magnitude in the image can take values in the range \([0, 1]\). The test pattern and the noise are denoted \( \mu \) and \( \nu \) respectively.

\[
dB_{\infty} = \frac{90.5}{345} \mu \tag{3.8}
\]

\[
dB_{20} = \frac{255}{345} \left( \frac{90.5}{255} \mu + \frac{1}{10} \nu \right) \tag{3.9}
\]

\[
dB_{10} = \frac{255}{345} \left( \frac{90.5}{255} \mu + \frac{1}{10} \nu \right) \tag{3.10}
\]

\[
dB_{0} = \frac{255}{345} \left( \frac{90.5}{255} \mu + \nu \right) \tag{3.11}
\]

The curvature direction test pattern corresponding to fig. 3.4 The test pattern is composed by three different images: two curvature direction test patterns and one image controlling the choice of pattern. The control image is a \((r, \phi)\) chess-pattern defined by

\[
\begin{cases} 
1 & \text{if } \cos(36 \cdot \arg(\xi'_1 + i\xi'_2)) \cdot \sin(36 \cdot \ln(r)) > 0 \\
0 & \text{otherwise} 
\end{cases} \tag{3.12}
\]

The two test patterns are defined as

\[
\exp(i \cdot 128\pi r) \tag{3.13}
\]

and

\[
\exp(i \cdot 128\pi \xi'_1) \tag{3.14}
\]

where the spatial coordinate \( \xi'_1 \) is normalized such as \( \xi'_1 \in [-1, 1] \). The correct result for these test patterns are vectors directed as \( \xi'_1 + i\xi'_2 \) for the first pattern and a constant vector directed as \( 1 + 0i \) for the second pattern. The performance measures are computed for points in the image where the polar spatial coordinate \( r \) fulfills \( r \in [0.1, 1] \).

Note that this type of image has all types of abrupt orientation/direction changes.
Performance measures The error measures have been chosen in a way which ensures that if $f_2^{ref} = f_2^{est}$ (reference and estimated value respectively) for all points $(\xi_1, \xi_2)$ where $f_2^{ref} \neq 0$, then both measures will result in the same value.

The weighted angular error is computed as

$$
\arccos \frac{\sum_{x,y} d(\xi_1, \xi_2)}{\sum_{x,y} |d(\xi_1, \xi_2)|}
$$

(3.15)

where $d(\xi_1, \xi_2)$, the difference vector, is

$$
d(\xi_1, \xi_2) = |f_2^{ref}(\xi_1, \xi_2)||f_2^{est}(\xi_1, \xi_2)| \exp(j(\arg(f_2^{ref}(\xi_1, \xi_2)) - \arg(f_2^{est}(\xi_1, \xi_2))))
$$

(3.16)

The angular error is computed in a similar way.

$$
2 \arcsin \sqrt{\frac{\sum \sin^2(\arg(d(\xi_1, \xi_2))/2)|f_2^{ref}|}{\sum |f_2^{ref}|}}
$$

(3.17)

The two angular measures have a behaviour similar to the standard deviation and can, since they are equal if the estimated certainty equals the reference certainty, be compared to each other.
Chapter 4

Application Examples: Texture Analysis and Image Enhancement

The two previous chapters have introduced a hierarchical curvature estimation method and discussed the bottom-up processing within the hierarchical structure. The curvature is by itself useful as a feature for many computer vision and image processing tasks and this chapter presents some examples of applications where the curvature feature has been incorporated into existing algorithms, namely texture analysis (Knutsson and Granlund [61]) and image enhancement (Knutsson, Wilson and Granlund [63]). This exemplifies the top-down (feedback) processing within the hierarchical structure.

4.1 Texture Analysis

The algorithm described in [61] consists of the following steps:

1. Estimate local orientation and local frequency. This gives a pixel-based description.

2. Transform the pixel-based description into a description in terms of texture elements (texels).

3. Use the texel-based description as input to a classification algorithm, e.g. Maximum-Likelihood.

The method is able to discriminate textures which differ in directionality (orientation) and coarseness (frequency). Fig. 4.1 contains six different synthetic textures. Textures A and F are discriminated from the other textures by a different frequency content, while textures B and C are discriminated by the orientation feature.

Note that the textures given in fig. 4.1 are archetypes of the textural features described by the method. With this method it is possible to classify textures that are considerably more difficult to discriminate. See [61] for examples of classification of natural textures.

The textures D and E in fig. 4.1 can, however, not be discriminated solely on frequency and orientation since D and E have identical orientation and frequency.
Figure 4.1: Six different synthetic textures

statistics. These two textures are archetypes of the textures described by the curvature direction. This textural feature can be said to be of second order (since it is estimated by a two-step procedure) and bridges some of the gap to high-level descriptions of textures (e.g., Julesz and Bergen [56]).

The result of classification of the textures with the use of an unsupervised minimum distance algorithm (Mantaras and Aquilar-Martin [33]) is displayed in fig. 4.2. Each grey level represents one of the clusters obtained by the classification algorithm. Orientation, frequency and curvature direction was used as features in this classification. The fuzzy borderline is an effect of the second step in the texture analysis, the transfer from pixel to texel description, which was achieved by vector averaging. The use of more sophisticated techniques such as multi-resolution segmentation (e.g., Wilson and Spann [101]) may overcome this difficulty. The small clusters located on the borderline are not displayed in fig. 4.2.

Another example of texture discrimination with the curvature direction estimate is given in fig. 4.3-4.4 where the difference between the two textures is more subtle. The other curvature algorithms can also be used for texture analysis. See [9, fig. 4.19-4.20] for an example of discrimination of synthetic textures with the use of the curvature magnitude measure (chapter 2.2.3) in combination with the curved/straight measure (chapter 2.2.2).

Finding natural textures where the curvature estimates are of use is not as easy as finding textures where the more basic features orientation and frequency play an important role in the discrimination. More sophisticated methods for the transfer from pixel to texel description as well as new classification methods may change that. It is, however, easy to demonstrate that the relationship between the orientation
Figure 4.2: The classification of the textures in fig. 4.1

Figure 4.3: Two synthetic textures not easily discriminated. Hint: the second texture is found in the centre of the image in a circular region
Figure 4.4: The classification of fig.4.3 using the curvature direction estimate. The true border between the textures is superimposed.

estimate and the curvature direction estimate is useful. Recall eq. (2.27) which holds true for all neighbourhoods constituted by curves, but not for line ends and discontinuous orientation changes. This inspires to the following texture measure

$$\sqrt{||f_1(\xi_1, \xi_2)|| ||f_2(\xi_1, \xi_2)|| \exp(i[2 \arg(f_2(\xi_1, \xi_2)) - \arg(f_1(\xi_1, \xi_2))])}$$

where \((\xi_1, \xi_2)\) denotes the spatial coordinates, \(f_1\) is the complex-valued orientation estimate and \(f_2\) is the curvature direction measure. This measure gives a vector directed as \(\pi\) for neighbourhoods with continuous orientation changes and should be computed prior to the pixel to texel description transfer.

Three different natural textures from Brodatz [28] are used: plastic bubbles (fig. 4.5); straw (fig. 4.6); and Herringbone weave (fig. 4.7). ‘Bubbles’ is a good example of continuous orientation changes, while ‘Straw’ exemplifies discontinuous orientation changes. The ‘Weave’ textures has both kinds, but at different scales. The orientation changes of the threads are smooth, while the orientation changes of the fields are abrupt. Using eq. (4.1) at fine and coarse resolution gives different estimates, one ‘micro-texture’ and one ‘macro-texture’. The texel descriptions of the different textures are visualized as angular histograms in fig. 4.8. Note that the ends of the x-axis wrap around.

4.2 Image Enhancement: Grey Level Images

Image enhancement is an important field of image processing. It consists of a collection of techniques that seek to improve the visual appearance of an image, or to
Figure 4.5: The ‘plastic bubbles’ image

Figure 4.6: The straw image
recover an ideal high quality image from a degraded version. At present there is no unifying theory of image enhancement, since it is difficult to define a criterion characterizing a "good quality" image. The problem is aggravated due to the fact that the quality of an image is determined by the not so well understood human visual system.

One of many image enhancement techniques is adaptive anisotropic (directed) filtering, which claims that the existence and direction of locally one-dimensional structures are important for the visual appearance of an image. One-dimensional structures are typically lines and edges, and that such structures are important cues is a widely spread opinion. This is the motivation behind the algorithm described in [63]. The next section describes this algorithm in some detail before the proposal of a similar algorithm which also takes the curvature estimate into account.

4.2.1 The Knutsson, Wilson and Granlund Algorithm

The general idea is to estimate the location, orientation and the visual significance of the edges from the noisy image and to use this as context control for an adaptive linear filter. The human eye and visual system are much more sensitive to errors in spatially broad, flat, low frequency regions than at edges, thus the adaptive filter smooths over large regions in low frequency areas and smooths over very small regions in high frequency areas, e.g. near edges.

Furthermore, the effect of noise in the vicinity of a linear feature depends on its orientation relative to the feature. Noise in the same orientation may enhance detectability, and noise in the perpendicular orientation will reduce it. Thus the filter need to be adaptive depending on not only the frequency of the input, but
Figure 4.8: The angular distribution of the texture measure defined in eq. (4.1)
also on the orientation of the one-dimensional structure. The resulting filter will be an adaptive anisotropic filter. This is possible to achieve by interpolation of linear filters (e.g., appendix 2.C).

An algorithm having the features described above can be implemented according to the equations:

\[ f'(\xi_1, \xi_2) = LP + \alpha HP \]
\[ HP = \beta HP_{aniso} + (1 - \beta) HP_{iso} \]

where \( f'(\xi_1, \xi_2) \) is the grey level of the actual pixel in the enhanced image, \( LP \) is the output from a lowpass filter, \( HP \) is the output from an isotropic highpass filter, \( HP_{aniso} \) is the output from an oriented highpass filter (the orientation is given by a parameter \( \varphi_0 \)), \( \alpha \) is a weight factor, controlling the influence of high frequency components in the output and \( \beta \) is a weight factor, controlling how much of the high frequency filter that should be oriented.

The parameters \( \alpha \) and \( \beta \) will control the resulting filter shape. \( \alpha \) will control the size of the filter used. The larger value (between 0 and 1) \( \alpha \) is given, the smaller (spatially) the filter will be. \( \beta \) will control the amount of 'orientedness'. The larger \( \beta \) (in the interval 0 to 1), the more oriented the filter will be (fig. 4.9). The values of \( \alpha \) and \( \beta \) are estimated from the original image. This is done with the orientation estimation algorithm described in chapter 2.1. An overview of the enhancement algorithm is given in fig. 4.10.

Estimation of Context Information

The orientation algorithm extracts the location and orientation of lines and edges eq. (2.5) and the local energy (the scalar sum of the quadrature filters, computed with eq. (2.5) without the factor \( \exp(i2\phi_k) \)) in the image. These values are combined locally to a certainty measure of both the existence and the orientation of the one-dimensional structures. These quantities (orientation, certainty and energy) will control the values of \( \alpha, \beta \) and \( \varphi_0 \).

The certainty measure mentioned above is computed with relaxation, i.e., vector averaging. The relaxation is necessary to allow larger spatial regions to control the certainty of the orientation estimation. The inspiration to this is found in
mammalian vision, and the appearance of the enhanced images controlled by relaxed contextually data is preferable to using the output from the orientation algorithm directly.

When deciding whether a neighbourhood has an orientation or not, the local energy is not supposed to interfere. By normalizing the oriented energy with the total energy an energy independent certainty, \( C \), is achieved:

\[
C(\xi_1, \xi_2) = \frac{||h_a * f_1(\xi_1, \xi_2)||}{h_a * M_1(\xi_1, \xi_2)}
\]

(4.4)

where \( h_a \) is an average filter, \(*\) denotes convolution, \( f_1 \) is the complex-valued orientation estimate and \( M_1 \) is the local energy estimate. The interpolation angle, \( \varphi_0 \), is given in double angle representation from the vector average of the orientation estimation image (\( \arg(h_a * f_1) \)). The value of the parameter \( \alpha \), that controls the amount of high frequency components, is taken from the average of the energy image \( (h_a * M_1) \). The value of \( \beta \), 'the orientedness of the filter', is computed as \( 1 - \sqrt{1 - C} \).

The Filters Used for Enhancement

The algorithm requires a filter set from which it is possible to

- change the frequency characteristics of the interpolated filter
- interpolate anisotropic filters in arbitrary orientations

This is achieved by four linear filters: one isotropic lowpass, one isotropic highpass, and two anisotropic highpass filters. The filter function in the Fourier domain for the isotropic lowpass filter is

\[
F_{\text{LP}}(\rho) = \begin{cases} 
\cos^2(\frac{\pi \rho}{1.8}) & \rho \leq 0.9 \\
0 & \rho > 0.9 
\end{cases}
\]

(4.5)
where \( \rho \) denotes the frequency. The radial part of the highpass filters is

\[
F_{HP}(\rho) = \begin{cases} 
1 - \cos^2\left(\frac{\pi \rho}{1.8}\right) & 0 \leq \rho < 0.9 \\
1 & 0.9 \leq \rho < \pi - 0.9 \\
\cos^2\left(\frac{\pi (\rho - \pi + 0.9)}{1.8}\right) & \pi - 0.9 \leq \rho < \pi
\end{cases}
\]  
(4.6)

The frequency \( \rho = \pi \) corresponds to the maximum image frequency (the Nyquist frequency). The values, 0.9 and 1.8, of the constants in eq. (4.5) and (4.6) are chosen under the demand that the filters must be possible to realize spatially on a \( 15 \times 15 \) grid.

The angular parts of the highpass filters are

\[
\begin{align*}
F_{\text{iso}}(\varphi) &= 1 \\
F_{\text{cos}}(\varphi) &= \cos 2\varphi \\
F_{\text{sin}}(\varphi) &= \sin 2\varphi
\end{align*}
\]  
(4.7)

Using trigonometric formulas and these filter functions in (4.2) and (4.3) gives

\[
\text{OUT} = LP + \alpha[H_{\text{iso}} + \frac{1}{2}\beta(H_{\text{cos}} \cos 2\varphi_0 + H_{\text{sin}} \sin 2\varphi_0 - H_{\text{iso}})]
\]  
(4.8)

where \( LP, H_{\text{iso}}, H_{\text{cos}} \) and \( H_{\text{sin}} \) are the outputs from respective filters. This gives anisotropic filters with an angular variation of \( \cos^2(\varphi - \varphi_0) \).

The behaviour of the interpolated filter for different values of \( \alpha \) and \( \beta \) is visualized in fig. 4.11. In this figure the final filtershape is plotted in the frequency domain.

The behaviour of the algorithm can be summarized as:

- Neighbourhoods with very low energy are lowpassed.
- Neighbourhoods with high energy will, if the orientation in the neighbourhood is certain, be enhanced with an anisotropic filter.
- If the orientation is not specific but the energy is high in a neighbourhood, typically around a corner, the algorithm will enhance the image using an isotropic filter.

### 4.2.2 Curvature Modifications to the Algorithm

The algorithm described above has good performance for images with a signal to noise ratio \( \geq 0 \) dB, and it is used successfully in applications (e.g., medical [46]). This fact illustrates that the risk of introducing artifacts is very small. The risk does, however, exist and the curvature estimate can be used to decrease this risk. Other modifications are also possible. A multi-resolution extension of the original algorithm is given in Clippingdale and Wilson [29].

The algorithm has a number of parameters to tune the performance, e.g., increase the tendency to use anisotropic filtering. Artifacts may appear if one exaggerates the anisotropic part for neighbourhoods with relatively uncertain orientation estimates, especially if the procedure is iterated. Gaps in lines can for instance be bridged (which, depending on the application, can be both a bug and a feature) and curves
Figure 4.11: The filter shape for different values of $\alpha$ and $\beta$ in the Fourier domain.
with sharp bends may be straightened out to lines. The values of the tuning parameters differ from application to application but are stable within an application. The initial selection of the parameter values is a careful process, where the objective is to recover as much as possible of the image structure without introducing artifacts.

The introduction of the curvature information decreases the risk of 'over-enhancement'. The previous described algorithm uses three different neighborhood models. A fourth model will now be added; the curved line model. An illustration of the local Fourier spectra of a neighborhood constituted by a curve is given in fig. 2.7. The enhancement algorithm should be able to adapt the filter shape to this butterfly shape.

The following formulation of the new grey-level image enhancement algorithm keeps the algorithmic structure of fig. 4.10 intact: Supplement eq. (4.2)–(4.3) with a context parameter \( \gamma \), defining the shape of the anisotropic filter. This requires a more extensive context information computation to determine \( \gamma \) (essentially an addition of the curvature direction algorithm), and the use of a larger number of filters for the interpolation procedure.

An angular filter shape not suppressing the important parts of the Fourier spectrum in fig. 2.7 is achieved by

\[
F(\varphi) = 1 - \sin^{2A}(\varphi - \varphi_0)
\]

for parameter values \( A > 1 \). Note that one could also do the opposite, change the filter shape to more orientation selective shapes, e.g. \( \cos^8 \phi \). Using filters with the angular function given by \( \cos N\varphi \) has, for large values of \( N \), two problems.

1. The spatial size will, for a fixed radial frequency function, increase with \( N \).
2. The Fourier domain specification of the filter requires a denser sampled grid, since for each value of \( \rho \) where the radial weight function \( \neq 0 \) there must exist at least \( 2N \) samples on the circle with radius \( \rho \).

The current implementation of the algorithm uses filters of order up to \( N = 8 \). Fig. 4.12 gives some examples of filter shapes for different choices of \( \gamma \).

**Estimation of Curvature Context**

The curvature direction algorithm extracts the location and tangent direction of the curves (eq. (2.14)) and the local energy (the scalar sum of the quasi-quadrature filters, i.e. exclude the factor \( e^{i\theta_k} \) in eq. (2.14)) in the orientation image. These values are combined locally to an estimate of the context parameter \( \gamma \) describing the certainty measure of the existence and the direction of the curved structure. This is done in the same manner as the computation of \( C \) in eq. (4.4), i.e. by computing the quotient between the vector average of the curvature vectors and the scalar average of the local energy measures. The quotient is then mapped by a suitable function (e.g. a sigmoid) to the estimate of \( \gamma \).

The curvature context estimate can also be used to influence both \( \beta \) and \( \varphi_0 \). If the curvature estimate is more certain than the orientation estimate, then it may be better to use the curvature certainty to determine \( \beta \) and the curvature direction \( \varphi_0 + \pi/2 \) to determine \( \varphi \).
Figure 4.12: The influence of $\gamma$ on the anisotropic filter shape for $\gamma = 0, 0.25, 0.5, 0.75$ and 1 (eq. (4.10)). The $\gamma = 0$ curve is the one with a narrow top and an almost flat bottom. The scale of the x-axis is degrees.
The final choice of weight functions for the combination of context information should be guided by heuristics for the specific application where performance evaluations like the ones in chapter 3 may play an important role.

The Filters Used for Enhancement

The $\gamma$-value defines the anisotropic filter shape. A number of base shapes is distributed on a ' $\gamma$-axis', and linear interpolation is used to create the filter shape for values of $\gamma$ between the basis shapes. The interpolation is done in three steps.

1. Filter the image with one isotropic lowpass filter and a number of highpass filters with different angular frequency functions: 1, $\cos 2\varphi$, $\sin 2\varphi$, $\cos 4\varphi$, ...

2. Select the two basis shapes adjacent to the current $\gamma$ and interpolate the filter response of these basis shapes for the direction $\varphi_0$ from the filter response.

3. Use linear interpolation to get the filter response for the shape corresponding to $\gamma$. This is the filter response to be used in eq. (4.3).

The current implementation with filters having angular frequency functions up to $\cos 8\varphi$ uses the following basis shapes:

\[
\begin{align*}
F_{\gamma=0}(\varphi) &= \cos^8(\varphi - \varphi_0) \\
F_{\gamma=1/6}(\varphi) &= \cos^6(\varphi - \varphi_0) \\
F_{\gamma=1/3}(\varphi) &= \cos^4(\varphi - \varphi_0) \\
F_{\gamma=1/2}(\varphi) &= \cos^2(\varphi - \varphi_0) \\
F_{\gamma=2/3}(\varphi) &= 1 - \sin^4(\varphi - \varphi_0) \\
F_{\gamma=5/6}(\varphi) &= 1 - \sin^6(\varphi - \varphi_0) \\
F_{\gamma=1}(\varphi) &= 1 - \sin^8(\varphi - \varphi_0)
\end{align*}
\]

(4.10)

Interpolation formulas are given in appendix 4.A.

The influence of the curvature addition to the algorithm can be summarized as:

- The use of anisotropic filtering in curved neighbourhood is increased.
- Neighbourhoods with low curvature will be enhanced with a narrow anisotropic filter.
- Neighbourhoods with high curvature will be enhanced with a broad anisotropic filter.

The difference between the enhancement result obtained by the original algorithm (chapter 4.2.1) and the extended algorithm is small for most neighbourhoods, but the result in fig. 4.13 illustrates the usefulness of the addition.

4.3 Image Enhancement: Vector Images

The generalization of the image enhancement algorithm to complex-valued images is quite straightforward if a polar separable approach is chosen. The local curvature model (eq. (2.10) and fig. 2.7) is easily incorporated into the following scheme.
The magnitude part of the local neighbourhood is processed by the algorithm described in chapter 4.2.1 to produce estimates of $\alpha$, $\beta$ and $\varphi_0$.

The curvature direction $t$ and magnitude $\kappa$ is estimated by the algorithms described in chapter 2.2.

The precomputed filters are modulated according to eq. (2.10). The modulation frequency is controlled by $\kappa$ and the direction is determined by $t$.

The filter responses of the modulated filters are combined according to the values of $\alpha$, $\beta$ and $\varphi_0$.

The algorithm is visualized in fig. 4.14 and should be compared to the corresponding figure (fig. 4.10) describing the grey-level algorithm.

It is also necessary to incorporate one extra context parameter, $\delta$, describing the certainty of the estimates $\kappa$ and $t$. Writing the algorithm in equations gives

$$f_1'((\xi_1, \xi_2) = \delta V_{\text{mod}}(\xi_1, \xi_2) + (1 - \delta)V_{\text{enh}}(\xi_1, \xi_2)$$

where $V_{\text{mod}}$ is the output from the computation of eq. (4.2) and eq. (4.3) using modulated filters and $V_{\text{enh}}$ is the output from computation with unmodulated filters. The addition of eq. (4.11) introduces filtering with the unmodulated filters as an additional item in the list above. This addition is included in fig. 4.14.

A result obtained by the enhancement algorithm is given in fig. A.8 together with the original data and the result obtained by vector averaging. The complex-valued input data contains a rotating vector field of constant magnitude together
with Gaussian white noise. The intensity of the noise increases to the right. Vector averaging does not extract the rotating pattern, while the enhancement algorithm performs very well.

4.4 Conclusion

The usefulness of the curvature estimates has been illustrated in a few examples to demonstrate the simplicity of incorporating curvature as a useful feature in image processing and computer vision tasks. The given examples are not full-fledged applications yet. Heuristics (or a theory) for the different applications should be added to get the most out of the presented methods. The material presented in this chapter gives possibilities such as cascading operations, e.g., to use vector enhancement on the orientation data before enhancing the grey level image.
Appendices

4.A Interpolation Formulas for Basis Shape Interpolation

A filter with an angular function of type $\cos^N(\varphi - \varphi_0)$ (where $N$ is an even number) is obtained by

$$S_N \left( w_{N0} H_{P_{iso}} + \sum_{k=1}^{N/2} w_{Nk} (H_{P_{cosk}} \cos 2k\varphi_0 + H_{P_{sink}} \sin 2k\varphi_0) \right)$$

(4.12)

where $H_{P_{iso}}$ is an isotropic bandpass filter, $H_{P_{cosk}}$ is a filter with identical radial frequency function and the angular frequency function $\cos(2k\varphi)$, $H_{P_{sink}}$ has the angular frequency function $\sin(2k\varphi)$, and the scale factor $S_N$ is given by the $n$-th position in the vector (indexed from zero):

$$\left( 1 \ 1/2 \ 1/8 \ 1/32 \ 1/128 \ 1/512 \ 1/2048 \ 1/8192 \ \ldots \right)^T$$

(4.13)

The values of $w_{Nk}$ are given by the element at row $N$ and column $k$ in the matrix (indexed from zero):

$$\left( \begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
10 & 15 & 6 & 1 & 0 & 0 & 0 & 0 \\
35 & 56 & 28 & 8 & 1 & 0 & 0 & 0 \\
126 & 210 & 120 & 45 & 10 & 1 & 0 & 0 \\
462 & 792 & 495 & 220 & 66 & 12 & 1 & 0 \\
1716 & 3003 & 2002 & 1001 & 364 & 91 & 14 & 1 \\
\end{array} \right)$$

(4.14)
Part II

3D — Volume and Time Sequence Processing

Ten sir, nine sir,
Eight sir, seven sir,
Six, five, four sir,
Three, two, one!

Tenser, said the tensor,
Tenser, said the tensor,
Tension, apprehension, and dissention have begun.

"THE DEMOLISHED MAN"— Alfred Bester
Chapter 5

Estimation, Representation and Visualization of 3D Orientation

The analysis of volumes and time sequences are both of great interest since there exists a rich variety of applications with this type of data. Volumes and time sequences are of the same dimensionality, and the algorithms presented here are essentially the same in both cases. This chapter describes the 3D orientation estimation algorithm. The next chapter introduces the 3D curvature estimation algorithms. The following chapter concerns time sequence analysis, where the algorithms are customized to utilize the fact that the third dimension is constituted by time.

Finding a suitable representation of the orientation of 3D surfaces may seem to be a trivial problem. This is, however, not the case due to the fact that orientation is defined modulo $\pi$. This discontinuity is taken care of in 2D by the use of the ‘double angle’ representation, i.e., an orientation $\phi$ is represented by a vector directed as $2\phi$ instead. No such elegant trick is available for 3D. It is demonstrated in Knutsson [65] that the use of a higher dimensional representation space is required, and a 5D-vector representation is proposed. There are some problems with the inverse mapping from 5D back to 3D and an alternative representation using generalized vectors, tensors, is described in Knutsson [60].

5.1 Using Tensors to Represent 2D Orientation

An alternative representation of local orientation is given by the orientation tensor

$$\frac{1}{x} \begin{pmatrix} x_1^2 & x_1 x_2 \\ x_1 x_2 & x_2^2 \end{pmatrix}$$

(5.1)

where $\text{arg}(x_1 + ix_2)$ corresponds to the direction of the gradient and $x = \sqrt{x_1^2 + x_2^2}$. This representation contains more information than the vector representation of orientation and has many of its virtues such as continuity (see Knutsson [60] for a more detailed description). The eigenvalues $\lambda_i$ and eigenvectors $e_i$ of $T$ give information regarding the local orientation. The eigenvector $e_1$ of the largest eigenvalue corresponds to the value of the dominant orientation. The relationship between the eigenvalues determines exactly how dominant that orientation is. For example, a one-dimensional neighbourhood has $\lambda_1 \gg \lambda_2$. The tensor representation is obtained...
in the same manner as the vector representation (eq. (2.5)), i.e.
by summation:

\[ T = \sum_{k} q_k (T_k - \alpha I) \]  

(5.2)

where \( q_k \) again denotes the magnitude of the filter response. \( T_k = \mathbf{n}_k \mathbf{n}_k^T \) denotes the
direction of the filter expressed in the tensor representation, \( I \) is the unity tensor
and \( \alpha \) is a constant.

It is desirable to convert the curvature direction algorithm to accept tensor data
input instead of complex valued input. This is easily done since there exists a
mapping from the tensor orientation representation to the vector orientation repre­
sentation. This transfers the tensor \( T \) to the complex number \( f_1 \) according to (see
appendix 5.A)

Recall that the curvature direction algorithm consists of a vector summation of
filter outputs (eq. (2.14)) obtained by convolution of the complex-valued orientation
image. The tensor image represents orientation in another way, but the vector sum­
ination and the filters remain the same. The only modification is the computation
of the magnitudes of the filter outputs. They should be computed as if the input
data consisted of ‘double angle’ orientation vectors. This is accomplished by using
eq. (5.3) to express the formula for complex convolution.

\[ q = (h_e + ih_o) * (x_1^2 - x_2^2 + i2x_1x_2) \]  

(5.4)

where \( h_e \) and \( h_o \) are the real and imaginary parts of the filter (in this case the even
and odd parts of the quasi-quadrature filter) and \( q \) denotes the complex-valued
filter output. The magnitude of the filter output, \( q \), is then computed as

\[ q = \sqrt{(h_e * (x_1^2 - x_2^2) - h_o * (2x_1x_2))^2 + (h_e * (2x_1x_2) + h_o * (x_1^2 - x_2^2))^2} \]  

(5.5)

This implies that the curvature direction algorithm can be implemented by filtering
of the tensor image representing orientation.

### 5.2 3D Orientation Estimation and Representation

It is easy to generalize the orientation algorithm to 3D when tensors are used. The
3D orientation is then expressed as

\[ \frac{1}{x} \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 \\ x_1x_2 & x_2^2 & x_2x_3 \\ x_1x_3 & x_2x_3 & x_3^2 \end{pmatrix} \]  

(5.6)

where \( (x_1, x_2, x_3) \) is the gradient vector, i.e. the normal vector to the dominant plane
of the neighbourhood and \( x = \sqrt{x_1^2 + x_2^2 + x_3^2} \). The representation is obtained by
eq. (5.2) with the only difference being the use of 3D quadrature filters.

The eigenvalues and eigenvectors of \( T \) describe the local orientation of the neigh­
bourhood. Three particular cases exist for the normalized tensor \( \hat{T} \) (\( \sum i_{ij}^2 = 1 \)):
This case corresponds to a neighbourhood that is perfectly planar, i.e., is constant on planes in a given orientation. The orientation of the normal vectors to the planes is given by \( \hat{e}_1 \).

- \( \lambda_1 = \lambda_2 = 1/\sqrt{2} \); \( \lambda_3 = 0 \);
  
  This case corresponds to a neighbourhood that is constant on lines. The orientation of the lines is given by \( \hat{e}_3 \).

- \( \lambda_1 = \lambda_2 = \lambda_3 = 1/\sqrt{3} \);
  
  This case corresponds to an isotropic neighbourhood.

All neighbourhoods can be expressed as linear combinations of these three cases.

A detailed presentation of the algorithm with proofs and a discussion of the benefits of the algorithm can be found in Knutsson [60]. Appendix 5.B contains the details concerning implementation.

### 5.3 Accuracy of the Orientation Estimate

This section motivates the use of quadrature filters for 3D orientation estimate by the evaluation of the performance on a synthetic test pattern. This \( 64 \times 64 \times 64 \) volume contains all possible 3D plane orientations for a wide frequency range. A traveller headed outwards from the centre and moving in a straight line would experience a sine wave with decreasing frequency.

Three instances of the test pattern were used for the evaluation, one without noise and two with Gaussian distributed noise added. The volumes with noise have a SNR of 10 dB and 0 dB respectively. Fig. 5.1 shows some 2D slices from the volumes.

The estimated orientation tensor \( \hat{T}^e \) (with tensor elements \( t^e_{ij} \)) was compared with the theoretical orientation tensor \( \hat{T}^f \) (with tensor elements \( t^f_{ij} \)) for all points in the volume. The comparison was done with the following two error estimates.

\[
\text{Total error} = \sqrt{\sum_{ij} (\hat{t}^e_{ij} - \hat{t}^f_{ij})^2}
\]

\[
\text{Weighted error} = \sqrt{\sum_{ij} (\|\hat{T}^e\|^2 \sum_{ij} (\hat{t}^e_{ij} - \hat{t}^f_{ij})^2) / \sqrt{\sum_{ij} \|\hat{T}^e\|^2}}
\]

where the \(^{-}\)-notation indicates that the tensor has been normalized, i.e., \( \sum_{ij} \hat{t}^2_{ij} = 1 \) and the norm of the tensor is computed as

\[
\|\hat{T}\| = \sqrt{\sum_{ij} \hat{t}^2_{ij}}
\]

The results obtained with quadrature filters are compared with the results produced by gradient filters, one of the more common methods for estimation of 3D
orientation (e.g. Zucker and Hummel [104], Monga and Deriche [74], Bigün et al [22], Jähne [55]). Both quadrature filters and gradient filters are of size $7 \times 7 \times 7$. The gradient estimate, $(x_1, x_2, x_3)^T$, is transformed into the tensor form by taking the outer product. The elements of the matrix are then scaled by a factor of $1/\sqrt{x_1^2 + x_2^2 + x_3^2}$. This gives a ‘tensor’ with a magnitude comparable with the one obtained by quadrature filters and enables the use of the same error estimates used for the quadrature filters (eq. (5.7) and (5.8)).

The error estimates are given in table 5.1. The implication from these results are clear. The advantage of the algorithm using quadrature filters is what one would expect, since one should be able to acquire a more robust estimate with twelve filters compared to three.

But one should not make premature conclusions from this single experiment. The frequency responses of the filters influence the noise suppression, and a more

<table>
<thead>
<tr>
<th>Test Pattern</th>
<th>Quadrature Filters</th>
<th>Gradient Filters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total error</td>
<td>Weighted Error</td>
</tr>
<tr>
<td>$\infty$ dB</td>
<td>1.47125</td>
<td>0.0903397</td>
</tr>
<tr>
<td>10 dB</td>
<td>2.58301</td>
<td>0.157799</td>
</tr>
<tr>
<td>0 dB</td>
<td>6.73242</td>
<td>0.407928</td>
</tr>
</tbody>
</table>

Table 5.1: The result of the comparison between gradient and quadrature filters using eq. (5.7) and (5.8)
thorough examination with a variety of frequency functions is needed. It is also possible to use other types of error measures. An extensive examination is, however, not the issue here. The only intention was to demonstrate that quadrature filters give accurate estimates of the 3D orientation.

Also note that it is possible to average the orientation tensor to get even more accurate estimates. This is a virtue of the representation and does not depend upon the filters used to achieve the initial estimates. Gradient filters in combination with this representation and averaging/relaxation have been used by Bigün et al [22] and Jähne [55].

5.4 Visualization of the Orientation Tensor

A good visualization technique is crucial for comprehension of the orientation information contained in the tensor. This section describes techniques for visualization of the orientation tensor. The last of the methods described here is useful for more than visualization. This interpretation technique is one of the fundamental building blocks utilized in the 3D generalization of the principal direction algorithm, described in the next chapter.

The six elements in the orientation tensor for the 3D case jointly describe the structure of the local neighbourhood. It is of course possible to visualize the tensor elements one at a time, but a method capturing the relationship between the elements of the tensor is preferable.

One method is to construct a matrix of images, where each image in the matrix displays the corresponding tensor element. Colour can be used to represent the sign of the values, e.g. green for positive and red for negative ones, while the magnitude is best represented by image intensity.

This approach displays all tensor elements simultaneously instead of visualizing one element at a time. It is nevertheless difficult to get a feeling for the local orientation structure since the values describing a specific point with this technique will be distributed in six different locations in the matrix image.

The eigenvalues and eigenvectors of the tensor give a full description of the local structure, and eigenvalue analysis taken together with computer graphics techniques suggest a means for good visualization of the local orientation:

1. Solve the eigenvalues and the eigenvectors of the tensor.

2. Render an ellipsoid using the eigenvalues to define the ellipsoid shape and the eigenvectors to choose the appropriate orientation in space.

This method visualizes the information locally in a very descriptive way, and is useful for the examination of specific points in the data set. However, there are two drawbacks when it comes to visualization of more global views of the data set. Firstly, the rendered ellipsoid requires more space for display than just a few pixels. This reduces the number of points made visible at the same time. Secondly, the rendered image, containing a subset of points in the data set, tends to be somewhat jumbled. Areas containing almost identical orientation values will not appear as

1 or nine, since the non-diagonal elements appear twice in the symmetric matrix.
smooth. Focusing on such areas and comparing neighbouring points overcomes this problem. Using colour to indicate the relationship between the eigenvalues is another remedy.

The visualization method proposed here combines the virtues of the two methods described above:

- It captures enough of the relationship between the tensor elements to give a good conception of the local orientation.
- The space requirement for rendering of the local structure of a point in the data set is small, i.e. three pixels.
- Smooth areas, where the orientation values are almost identical, appear as smooth areas.

The method is a generalization of the technique used to visualize 2D orientation. The 2D orientation estimate consists of a single complex number. The complex number is mapped to the colour circle, letting the magnitude determining the intensity and the argument determining the colour. There is no difference if the tensor representation introduced in this chapter is used. It requires only a preparatory mapping from the tensor to a complex number (Eq. (5.3)).

This mapping from a tensor to a complex number is a clue as to how to generalize the technique. In the same manner the 3D orientation tensor can be interpreted as three complex numbers.

\[
\begin{align*}
x_1^2 - x_2^2 + i2x_1x_2 \\
x_1^2 - x_3^2 + i2x_1x_3 \\
x_2^2 - x_3^2 + i2x_2x_3
\end{align*}
\]  

Intuitively the mapping corresponds to the following procedure. Take the normal vector \((x_1, x_2, x_3)^T\) of the local surface patch, make three different 2D projections of it, and represent each of the projections with the ‘double angle’-method.

The mapping of the 3D orientation tensor described by eq. (5.10)–(5.12) gives three complex-valued data sets which can be visualized using the mapping to the colour circle.

Positions in the data set where the projection fits will have high intensity and the colour will indicate the ‘2D-orientation’ with respect to the projection plane. Positions where data and projection do not match will have low intensity.

In fact, each of the complex values extracted from the orientation tensor has an equatorial region on the unit sphere, where the ‘sub-representation’ gives an adequate description of the orientation. There are also two ‘poles’, where the complex value has zero or low magnitude and the descriptive content is zero. However, this does not matter since it coincides with the equatorial regions of the two other complex valued ‘sub-representations’.

The above two ‘compact’ visualization techniques, i.e. the display of the tensor element images in a matrix and the display of the mappings into complex numbers are exemplified in fig. A.9 and A.10. The original data is a CT-volume consisting
of 100 slices, each of resolution $256 \times 256$. The volume contains a human skull from a cadaver embedded in a plastic material moulded into the shape of a human head. This volume was resampled to 150 slices to obtain approximately cubic voxels (fig. 5.2). The result obtained by the orientation algorithm is visualized with the two different methods in fig. A.9 and A.10.

5.4.1 Visualization of an Arbitrary Orientation Plane

The previous section described the visualization of the orientation tensor by projection of the representation into the $x_1x_2$-plane, $x_1x_3$-plane or $x_2x_3$-plane. The 2D-vectors (complex numbers) obtained by the projection of the orientation data are of ‘double angle’ type and they are suitable for visualization with colour and intensity representing the argument and magnitude of the 2D-vectors respectively.

This visualization scheme with ‘representation-projections’ can be generalized to arbitrary 2D-planes. Two different methods are described here.

The straight-forward method is to apply a rotation matrix, $Q$, on the orientation tensor, $T$, followed by the combination of the tensor elements into a complex number using the combination formula (5.10). This can be described by the following step by step procedure:

1. Compute the rotation matrix $Q$ which describes the rotation from the desired 2D-plane to the $x_1x_2$-plane. This can be done with standard methods, see e.g. Heise and MacDonald [18].
2. Rotate the tensor $T$ using

$$QTQ^{-1} = QTQ^T = T'$$

3. Combine the tensor elements of $T'$ into a complex number $f_1$ using

$$f_1 = t_{11}' - t_{22}' + 2it_{12}'$$

4. Visualize the complex numbers using the intensity to represent the magnitude and the colour to represent the argument.

The other method is to transform the combination rule instead. This is achieved by the following procedure:

1. Compute the rotation matrix $Q$ which describes the rotation from the desired 2D-plane to the $x_1x_2$-plane.
2. Rotate the vector $v = \begin{pmatrix} 1 & i \\ 0 & 0 \end{pmatrix}^T$ using $Q^{-1}$.

$$Q^{-1}v = Q^Tv = v'$$

3. Extract the complex number $f_1$ from the tensor using

$$v'^T Tv' = f_1$$

4. Visualize the complex valued data set using the intensity to represent the magnitude and the colour to represent the argument.

That these methods are equivalent is demonstrated in appendix 5.C.

5.5 Summary

The orientation representation using tensors proposed by Knutsson [60] has been presented. The 2D algorithms for curvature estimation have been modified to accept input data represented as tensors. The performance of the algorithm for 3D orientation estimation has been exemplified and methods for the visualization of the algorithm output have been discussed.

Appendices

5.A The Mapping from 2D Orientation Tensor to ‘Double Angle’ Vector

The orientation algorithm can be expressed both in vector and tensor form. Assume for example that four quadrature filters are used in the orientation estimation. Let
them be directed in the Fourier domain as \((1, 0, 0)^T, (1/\sqrt{2}, 1/\sqrt{2})^T, (0, 1, 0)^T\) and 
\((-1/\sqrt{2}, 1/\sqrt{2})^T\), i.e. \(\varphi_k = \pi(k - 1)/4\). The vector formulation is

\[
\mathbf{v} = \sum_{k=1}^{4} q_k \exp(i2\varphi_k) = (q_1 - q_3) + i(q_2 - q_4)
\]

(5.13)

The tensor formulation is

\[
\mathbf{T} = \sum_{k=1}^{4} q_k (T_k - \frac{1}{4} \mathbf{I})
\]

(5.14)

with

\[
T_1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}
\]

(5.15)

\[
T_2 = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}
\]

(5.16)

\[
T_3 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}
\]

(5.17)

\[
T_4 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}
\]

(5.18)

This results in

\[
\mathbf{T} = \begin{pmatrix} q_1 + \frac{1}{2} q_2 + \frac{1}{2} q_4 - \frac{1}{4} \sum q_k \\ \frac{1}{2} q_2 - \frac{1}{2} q_4 \\ \frac{1}{2} q_2 + q_3 + \frac{1}{2} q_4 - \frac{1}{4} \sum q_k \end{pmatrix} \begin{pmatrix} \frac{1}{2} q_2 - \frac{1}{2} q_4 \frac{1}{2} q_2 + q_3 + \frac{1}{2} q_4 - \frac{1}{4} \sum q_k \end{pmatrix}
\]

(5.19)

The transfer from tensor \(\mathbf{T}\) to vector \(\mathbf{v}\) is done by

\[
(t_{11} - t_{22}) + 2t_{12} = (q_1 - q_3) + i(q_2 - q_4) = \mathbf{v}
\]

(5.20)

### 5. B How to Implement the Orientation Algorithm

The algorithm requires a number of precomputed quadrature filters. The filters should be evenly spaced in one half of the Fourier space. In three dimensions there exists a limited number of ways to distribute the filters in a fully symmetrical fashion. The candidates are given by the five Platonic polyhedra, where only the icosahedron and the dodecahedron have a sufficient number of vertices. The minimum number of quadrature filters required for orientation estimation in 3D is 6, where the filters are directed as the vertices of a hemiicosahedron.

The 6 normal vectors are thus given by:

\[
\mathbf{n}_1 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \quad \mathbf{n}_2 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad \mathbf{n}_3 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} 1 + \sqrt{5} \\ -2 \end{pmatrix}, \quad \mathbf{n}_4 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \quad \mathbf{n}_5 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} 0 \\ 1 + \sqrt{5} \\ 2 \end{pmatrix}, \quad \mathbf{n}_6 = (10 + 2\sqrt{5})^{-1/2} \begin{pmatrix} 0 \\ 1 + \sqrt{5} \\ -2 \end{pmatrix}
\]

(5.21)

The procedure for the design of a quadrature filter is
Figure 5.3: The icosahedron used to define the Fourier domain direction of the 3D quadrature filter set.

1. Select an appropriate size for the frequency volume to be used in the Fourier domain filter design. The size should be at least $7 \times 7 \times 7$.

2. Given the Fourier domain direction of the filter, $\hat{n}_k$, and your favourite bandpass function, $H_p(u)$, compute

$$
\begin{align*}
H_k(u) &= H_p(u)(\hat{u} \cdot \hat{n}_k)^2 & \text{if } u \cdot \hat{n}_k > 0 \\
H_k(u) &= 0 & \text{otherwise}
\end{align*}
$$

(5.22)

for the frequency volume. The coordinate vector is denoted by $u = (u_1, u_2, u_3)$, $\hat{u}$ is a unit vector directed as $u$, and $u = \sqrt{u_1^2 + u_2^2 + u_3^2}$.

3. Convert the frequency volume to the spatial domain, either by a straightforward 3D-DFT or by use of an optimization technique (Knutsson [64]).

The resulting filter is complex-valued with the real part being even and the imaginary part being odd. This procedure is used to obtain the six complex-valued quadrature filters.

It is easy to implement the orientation algorithm with these precomputed filters. The algorithm is expressed by eq. (5.2) but a more detailed description is given here.

1. Convolve the input data with the six complex-valued filters, i.e. perform twelve scalar convolutions.

2. Compute the magnitude of each complex-valued filter by

$$
q_i = \sqrt{q_{ie}^2 + q_{io}^2}
$$

where $q_{ie}$ denotes the filter response of the real (and even) part of filter $i$ and $q_{io}$ denotes the filter response of the imaginary (and odd) part of filter $i$.

3. Compute the tensor $T$ by eq. (5.2), i.e.

$$
T = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{12} & t_{22} & t_{23} \\ t_{13} & t_{23} & t_{33} \end{pmatrix}
$$

84
where
\[
\begin{align*}
t_{11} &= A(q_1 + q_2) + B(q_3 + q_4) - S \\
t_{22} &= A(q_3 + q_4) + B(q_5 + q_6) - S \\
t_{33} &= A(q_5 + q_6) + B(q_1 + q_2) - S \\
t_{12} &= C(q_3 - q_4) \\
t_{13} &= C(q_1 - q_2) \\
t_{23} &= C(q_5 - q_6)
\end{align*}
\]
where
\[
\begin{align*}
S &= \frac{1}{5} \sum_{k=1}^{6} q_i \\
A &= \frac{4}{10 + 2\sqrt{5}} \\
B &= \frac{6 + 2\sqrt{5}}{10 + 2\sqrt{5}} \\
C &= \frac{2 + 2\sqrt{5}}{10 + 2\sqrt{5}}
\end{align*}
\]

5.C Arbitrary Orientation Plane Visualization

It is easy to demonstrate that the two methods for visualization of an arbitrary orientation plane are equivalent. First, show that

\[
v^T T v = t_{11} - t_{22} + 2it_{12}
\]

for \( v^T = (1 \ i \ 0) \). This is a straightforward computation

\[
\begin{pmatrix} 1 & i & 0 \end{pmatrix} \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{12} & t_{22} & t_{23} \\ t_{13} & t_{23} & t_{33} \end{pmatrix} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} = t_{11} - t_{22} + 2it_{12} \tag{5.23}
\]

Secondly, show that

\[
v'^T T v' = v^T T' v
\]

Using \( v' = Q^{-1} v, Q^{-1} = Q^T \) and \( T' = QTQ^{-1} \) gives

\[
v'^T T v' = vQTQ^{-1} v = v^T T' v
\]

which completes the proof.
Chapter 6

The Principal Direction Algorithm

The algorithm for estimation of the principal directions in 3D is essentially a straightforward generalization of the principal direction algorithm described in chapter 2, but there are some modifications due to the fact that both orientation and curvature are more complex events in 3D than in 2D. The algorithm modifications appear at the input to account for the use of a tensor to represent the orientation, and at the output, where some postprocessing is necessary for identification of the principal directions.

The next section reviews the concept of curvature in 3D and related work. The following sections describe the representation, the local neighbourhood model and the associated estimation algorithm.

6.1 The Problem

Definition and formulae concerning curvature originate in differential geometry [90, 68]. They are of course vital components in the design of a curvature estimation algorithm, although most of the groundwork for the methods in this thesis comes from the signal processing field.

A short review of the differential geometry formulae utilized in the derivation of the algorithm is given here as well as a few comments on some of the formulae not used.

A curve can be given in parameter form as

$$\xi_1 = y_1(s), \xi_2 = y_2(s), \xi_3 = y_3(s); \quad 0 \leq s \leq S \tag{6.1}$$

where $\xi_1, \xi_2$ and $\xi_3$ denotes the spatial coordinates, $y_i$ are scalar functions and $s$ denotes the arc length [90]. A curve parameterized by arc length has

$$\frac{dy}{ds} = \hat{t} \tag{6.2}$$

where $\hat{t}$ is a unit tangent vector.

The curvature $\kappa$ is defined as

$$k = \frac{d\hat{t}}{ds} = \kappa \hat{n} \tag{6.3}$$

86
where $k$ is the curvature vector and $\hat{n}$ is a unit vector perpendicular to $\hat{t}$. This is in accordance with the definition of curvature for 2D curves given in eq. (1.1). But 3D curves have an additional feature: the torsion $\tau$. While curvature measures the rate of change of the tangent, torsion measures the rate of change of the plane defined by the tangent and curvature vectors. Torsion and curvature together with the vectors $\hat{t}$ and $\hat{n}$ give a complete characterization of the curve. Another interesting shape feature of curves are the Darboux vector, $d$.

$$d = \tau \hat{t} + \kappa (\hat{t} \times \hat{n}) \quad (6.4)$$

A curve has mathematically no area. This fits badly with the discrete voxels of 3D data. Real world (digital volumes, that is) curves are better described as narrow surfaces and the formulae describing the curvature of surfaces are a better starting point for the design of curvature algorithms. This is not to say that curves are not important in 3D data. The optical flow of moving points (spatio-temporal curves) are for instance more reliable than the flow of moving lines (spatio-temporal surfaces). For that reason curves are discussed in chapter 7, but the curvature model in section 6.3 is formulated for surface geometry.

A surface can also be given in parameter form

$$\xi_1 = y_1(r_1, r_2), \quad \xi_2 = y_2(r_1, r_2), \quad \xi_3 = y_3(r_1, r_2); \quad R_1 \leq r_1 \leq R_1^+, \quad R_2^- \leq r_2 \leq R_2^+ \quad (6.5)$$

which can be compared to eq. (6.1). A difference is that the surface is not parameterized by arc length $s$. The element of arc is determined by the first fundamental form.

$$ds^2 = E dr_1^2 + 2F dr_1 dr_2 + G dr_2^2 = dy \cdot dy \quad (6.6)$$

where

$$E = \frac{dy}{dr_1} \cdot \frac{dy}{dr_1}, \quad F = \frac{dy}{dr_1} \cdot \frac{dy}{dr_2}, \quad G = \frac{dy}{dr_2} \cdot \frac{dy}{dr_2} \quad (6.7)$$

Another important entity is the second fundamental form.

$$edr_1^2 + 2f dr_1 dr_2 + gdr_2^2 = -dy \cdot d\hat{n} \quad (6.8)$$

where $\hat{n}$ now denotes a unit normal vector of the surface and

$$e = -\frac{dy}{dr_1} \cdot \frac{d\hat{n}}{dr_1}, \quad 2f = -(\frac{dy}{dr_1} \cdot \frac{d\hat{n}}{dr_1} + \frac{d\hat{n}}{dr_2} \cdot \frac{dy}{dr_2}), \quad g = -\frac{dy}{dr_2} \cdot \frac{d\hat{n}}{dr_2} \quad (6.9)$$

The quotient of the second fundamental form by the first is called the normal curvature (denoted $\kappa_n$) and describes the curvature of the surface for the tangent direction
are correct. There are two directions on the surface where \( \kappa_n \) obtains an extreme value, the principal direction of most curvature, \( t_1 \), and the principal direction of least curvature, \( t_2 \). These curvature directions are orthogonal. Euler’s theorem

\[
\kappa_n = \kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha
\]  

(6.10)

expresses the normal curvature in an arbitrary direction \( t \). \( \kappa_1 \) denotes the curvature in direction \( t_1 \) and \( \kappa_2 \) denotes the curvature in direction \( t_2 \). \( \alpha \) is the angle between \( t \) and \( t_1 \).

Differential geometry excels in the use of curvilinear coordinates and coordinate transforms. The principal directions \( t_1 \) and \( t_2 \) are useful to determine an orthogonal coordinate system, but special care must be taken in the neighbourhood of umbilic points, i.e., points on the surface with \( \kappa_1 = \kappa_2 \). The algorithms presented in this thesis use the fixed cartesian grid supplied with the input data. As a consequence, no extra precaution is necessary for umbilics, but methods to localize umbilic points are given in chapter 6.6.1 since this feature is one way to give a qualitative measure of local surface shape.

Another shape measure is the classification into elliptic, parabolic or hyperbolic surface patches.

- Elliptic points where the second fundamental form maintains the same sign for all directions, e.g., points on a sphere.
- Parabolic points where the normal curvature is zero in one direction, e.g., points on a cylinder.
- Hyperbolic or saddle points where the second fundamental form does not maintain the same sign for all directions.

This classification can be done by computation of Gaussian curvature

\[
K = \kappa_1 \kappa_2 = \frac{eg - f^2}{EG - F^2}
\]  

(6.11)

with \( K > 0 \) for elliptic points, \( K = 0 \) for parabolic points and \( K < 0 \) for hyperbolic points. An algorithm to determine the sign of the Gaussian curvature will be given in chapter 6.6.2. Another often used feature is the mean curvature.

\[
M = \frac{1}{2} (\kappa_1 + \kappa_2) = \frac{Eg - 2FF + eG}{2(EG - F^2)}
\]  

(6.12)

### 6.1.1 Related Work

Sander and Zucker [87, 88] were among the first ones to develop algorithms for the estimation of curvature features in ‘true’ 3D images. The principal curvatures and principal directions are computed by a relaxation process [54] which incorporates the differential geometric properties of 3D surfaces. The relaxation process is initialized with a thresholded gradient image [104].

Kehtarnavaz and deFigueiredo [58] propose an algorithm for curvature and torsion estimation on 3D curves, and characterizes the curve with the use of the Darboux vector (eq. (6.4)). A drawback is that the algorithm needs thresholded data.
Among the algorithms which describe 3D shape without using curvature estimation as the means can for example Terzopolis et al [91], Solina and Bajcsy [89] and Pentland [82] be mentioned.

Curvature estimation are also used on the ‘2 1/2-D’ data given by range images, see e.g. Vemuri et al [94], where the location of the surface is given a priori.

6.2 The Representation

The representation is now presented before the introduction of the neighbourhood model. The representation of principal directions of curvature is, in accordance with the representation of 3D orientation, a tensor $P$ which for a parabolic neighbourhood is given by

$$P = p \hat{i} \hat{m}^T = p \begin{pmatrix} t_1 m_1 & t_2 m_2 & t_3 m_3 \\ t_4 m_4 & t_5 m_5 & t_6 m_6 \\ t_7 m_7 & t_8 m_8 & t_9 m_9 \end{pmatrix}$$

(6.13)

where $p$ is a certainty measure, $\hat{i}$ is a unit vector oriented as the principal direction of curvature and $\hat{m}$ is a unit vector defining the ‘cylinder orientation’, see fig. 6.1. $\hat{i}$, $\hat{m}$ and $\hat{x}$, the surface normal, are perpendicular to each other.

The representation is continuous and rotation invariant (see chapter 6.A). Neighbourhoods of more than one principal direction of curvature are represented by

$$P = \sum_i p_i \hat{i}_i \hat{m}_i^T$$

(6.14)

i.e., each of the principal directions has its own $P$-matrix and the total representation is the sum of the $P$-matrices. Separation of the principal directions of curvature is done in a manner similar to the one used to find the dominant orientation from the orientation tensor $T$ (eq. (5.6)) and is described later in this chapter. A few comments of the choice of representation are given before the introduction of the local neighbourhood model and the algorithm.
At first it may seem more natural to use the gradient of the orientation tensor with \(3 \times 3 \times 3\) element as representation. However, such a representation would describe all types of changes in the neighbourhood and not only curvature. This is also true if one chooses to use the orientation representation and orientation estimation algorithm on the individual orientation tensor elements. Another choice would be to use only the principal direction \(\hat{\mathbf{t}}\) itself as representation. It is well-known from differential geometry [68, 90] that this gives problems with continuity. See fig. 6.2 for an example of two curvature patches with identical principal directions \(\hat{\mathbf{t}}\) but different ‘shape’ (and \(\hat{\mathbf{m}}!\)).

Finally, note that the representation given in eq. (6.13) is a generalisation of the representation of the curvature direction in 2D. The rotation axis \(\hat{\mathbf{m}}\) is in this case constrained to \(\begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^T\) and the principal direction of curvature is constrained to always have \(t_3 = 0\) which gives

\[
\mathbf{P}_{2D} = p \begin{pmatrix} 0 & 0 & t_1 \\ 0 & 0 & t_2 \\ 0 & 0 & 0 \end{pmatrix}
\]  

(6.15)

which can be interpreted as a complex number \(f_2 = t_1 + it_2\) (Compare with fig. 2.10).

### 6.3 The Model

The purpose of this section is to formulate a 3D neighbourhood model for curvature. The starting point is the model used to formulate the curvature direction algorithm in chapter 2. The 2D algorithm is based on the hypothesis of one-dimensionality, i.e., the assumption that a small neighbourhood contains at most a single event. Features of the neighbourhood can thus be described with relatively few parameters, e.g., one value to describe the local orientation, one value to describe the principal direction etc. The variety of possible 3D neighbourhoods is bigger, but the same principle can be used. This motivates the postulate that a local 3D neighbourhood contains at most one surface (or one curve).

There will of course be some neighbourhoods which do not fit this model and a good estimation algorithm should have a ‘graceful degradation’, e.g., a certainty
measure describing the model fit which decreases when the neighbourhood does not fit the model. Moreover, the algorithm should be able to estimate and describe both principal directions of curvature.

The 2D curvature model utilizes an isomorphism between the data representation of 2D orientation and complex numbers. The connection between the representation of 3D orientation (a symmetric $3 \times 3$ matrix) and complex numbers was given in chapter 5.4. A closer examination in appendix 6.B gives that the three complex numbers given by eq. (5.10)–(5.12) are sufficient for representation of orientation.

Furthermore, the one-dimensionality assumption, eq. (2.10) must be modified. One should be able to take both principal curvatures into account. However, it turns out that the mapping of the local orientation information into complex numbers results in neighbourhoods where the angular change can be characterized as one-dimensional even if there are more than one principal direction of curvature present. This is demonstrated in appendix 6.C.

The direction of angular change will be located in the 2D-plane spanned by the two principal directions. There are three complex-value descriptions of a neighbourhood and the directions of angular change only coincide for parabolic neighbourhoods. In fact, the directions of angular change for the three complex-valued descriptions are given by the columns of the matrix $P$ in eq. (6.14)! This is also demonstrated in appendix 6.C.

The discussion above can now be summarized in the following model of local 3D neighbourhoods:

1. Assume that the local neighbourhood contains at most one surface.

2. The orientation algorithm produces an orientation tensor description of the local neighbourhood.

3. The orientation tensor is interpreted as three complex numbers

   \[ x_1^2 - x_2^2 + i2x_1x_2 \]  \hspace{1cm} (6.16)

   \[ x_3^2 - x_1^2 + i2x_1x_3 \]  \hspace{1cm} (6.17)

   \[ x_2^2 - x_3^2 + i2x_2x_3 \]  \hspace{1cm} (6.18)

This produces three complex-valued descriptions of the 3D neighbourhood.

4. Such a complex-valued description of a surface neighbourhood is reasonably approximated by

   \[ r(\xi) \exp(2\pi i t \cdot \xi) \]  \hspace{1cm} (6.19)

   where $\xi$ denotes the spatial coordinate vector and $r$ denotes the magnitude function. The exponential function approximates the angular function with $t$ denoting the dominant direction of the angular changes (which for a parabolic neighbourhood coincides with the principal direction). The magnitude function defines whether or not a voxel is part of the surface. Fig. 6.3 gives an example of the angular function.

Observe that a linear model in terms of pixel/voxel coordinates is used while differential geometry utilizes (arbitrary) coordinate systems in terms of arc-length/surface-elements. Both the model and the estimation algorithm are local and will not violate
the definitions and theorems of differential geometry provided that the volume is sampled densely enough.

The shift theorem gives that the local Fourier transform of neighbourhoods fitting this model (eq. (6.19)) will be

$$F(u) = R(u - t)$$

(6.20)

where $u$ denotes the frequency coordinates and $R$ denotes the Fourier transform of the magnitude function $r$. The magnitude is a real-valued function which implies that $|R|$ will have its centre of gravity in the origin. Estimation of the principal directions of curvature in a 3D orientation tensor data set is thus analogue to estimation of the direction to the centre of gravity in the local Fourier spectra of the complex-valued projections of the tensor data. The core of the 3D algorithm is consequently identical to the 2D algorithm.

### 6.4 The Algorithm

The surface curvature model described motivates the following estimation algorithm:

1. Interpret the orientation tensor as three complex number data sets.

2. Estimate the centre of gravity in the local Fourier spectra for each of the complex-valued representations and form the principal direction tensor $P$ given in eq. (6.14).

3. Extract the principal directions from the three estimates of the direction to the centre of gravity.

The first step is trivial and the second step is a straightforward extension of the 2D curvature algorithm while the last step requires some discussion. But the presentation starts with the two first steps.
6.4.1 Estimation of the Centre of Gravity in the Local Fourier Spectra

The estimation is done by filtering of the complex-valued orientation representation (eq. (6.16)-(6.18)). The model formulation in section 6.3 implies that the centre of gravity for the local Fourier spectra of the orientation data is shifted away from the origin when the neighbourhood contains one (or two) curvature directions. A vector pointing in the direction of the centre of gravity is obtained by filtering with a number of filters where each filter is concentrated in one partition of the Fourier domain and the filters are spread equally. The vector is computed by

\[ \mathbf{b} = \sum_{k=1}^{K} q_k \mathbf{n}_k \]  

(6.21)

where \( q_k \) denotes the magnitude of the filter response and \( \mathbf{n}_k \) denotes the Fourier domain direction of filter \( k \).

The algorithm has many similarities with the 3D orientation estimation algorithm. One of them is the problem of distributing the precomputed filters equally over the Fourier domain. The difference is that the entire Fourier domain should be covered and that any of the five Platonic polyhedra can be used.

The use of quasi-quadrature filters gives rotation invariant estimation of the direction to the centre of gravity of the local Fourier spectrum. The quasi-quadrature filters used here are designed in the Fourier domain as polar separable filters. The name indicates a similarity to quadrature filters (eq. (5.22)) and the one difference is the angular function.

\[ \begin{align*}
H_\theta(u) &= \cos^2 \frac{\theta}{2} \\
\theta &= \arccos(\hat{u} \cdot \hat{n}_k)
\end{align*} \]  

(6.22)

Directing the filters as the vertices of an octahedron results in an easy implementation with relatively low computationally demands, since it is possible to reduce the six complex-valued filters to four scalar filters. The procedure for doing this with \( 6 \times 4 \) scalar convolutions is described in appendix 6.F. The Fourier domain directions of the filters are given in appendix 6.G.

The procedure for design of a quasi-quadrature filter is the same as for design of a quadrature filter (chapter 5.B) except for the different angular function (eq. (9.9))
and the different filter directions (eq. (6.78)). The quasi-quadrature filters are in accordance with the quadrature filters spatially complex-valued with the real part being even and the imaginary part being odd.

It is easy to implement the centre of gravity estimation part of the algorithm with the precomputed filters. The interpretation from orientation tensor to complex number can be incorporated in the formulation of the centre of gravity estimation. A step-by-step description of the algorithm is given in appendix 6.H. The proofs, motivations and discussion of the algorithm are given in appendix 6.E.

### 6.4.2 Extraction of the Principal Directions

The three vectors obtained by using eq. (6.21) are a mixture of the two principal directions of curvature. They are in fact related to the column vectors of the earlier proposed representation $P$. Denoting the vector produced by computing eq. (6.21) for the complex-valued orientation representation given by $x_1^2 - x_2^2 + i2x_1x_2$ with $(1,2)$ and so on gives

$$P = \begin{pmatrix} b_{(2,3)} & b_{(1,3)} & b_{(1,2)} \end{pmatrix}$$

(6.23)

It was stated earlier that the tensor $P$ gives a full description of the principal directions of curvature. The relationship between the principal direction $t_1$, and the centre of gravity estimates $b_{(i,j)}$ is simple when the neighbourhood is parabolic. The 3D orientation changes can for this type of neighbourhood be expressed as a rotation around an axis $m = (m_1 \ m_2 \ m_3)^T$. The rotation axis and the principal direction are perpendicular, i.e. $m \perp t_1$. The mapping from principal direction to centre of gravity estimate is described by

$$b_{(i,j)} = Xm_k t_1 \quad ; \quad i \neq j, \ i \neq k, \ j \neq k$$

(6.24)

where $X$ describes both the relationship between the filters used and the curvature $\kappa_1$ and also how certain the orientation estimates are.

Elliptic and hyperbolic neighbourhoods are slightly more complicated. Eq. (6.66) gives

$$\begin{cases} b_{(1,2)} = X(m_3 t_1 + m'_2 t_2) \\ b_{(1,3)} = X(m_2 t_1 + m'_3 t_2) \\ b_{(2,3)} = X(m_1 t_1 + m'_3 t_2) \end{cases}$$

(6.25)

Eq. (6.25) introduces a number of new variables. $t_1$ denotes the principal direction of most curvature, while $t_2$ denotes the principal direction of least curvature, $m$ is the rotation axis of $t_1$ and $m'$ is the rotation axis of $t_2$.

Separation of $t_1$ and $t_2$ is done with the Hotelling transform [39, p. 122–125], i.e. by computing the eigenvalues and eigenvectors of the matrix

$$B = b_{(1,2)}b_{(1,2)}^T + b_{(1,3)}b_{(1,3)}^T + b_{(2,3)}b_{(2,3)}^T.$$  

(6.26)

The eigenvalues and eigenvectors are easily computed with standard methods such as the Jacobi method (e.g. Press et al. [84]). The eigenvector $e'_1$ of the largest eigenvalue $\lambda'_1$ determines, apart from the sign, $t_1$ and the eigenvector $e'_2$ of the second largest eigenvalue $\lambda'_2$ determines, also apart from the sign, $t_2$. The third eigenvalue should for a well defined surface be close to zero.
The centre of gravity vectors corresponding to $e_g \text{t}_1$ are obtained by

$$b^{(i,j)} = (e'_1 \cdot b_{(i,j)}) e'_1$$  \hfill (6.27)

Note that

$$\begin{cases}
\lambda_1' &= k^2_1 \\
\lambda_2' &= k^2_2
\end{cases}$$  \hfill (6.28)

where $k_i = X \kappa_i$, i.e. the amount of curvature 'seen through' the frequency function of the filters used and the certainty of the orientation estimate. It should be possible to obtain an estimate of $\kappa_i$ by use of scale-space techniques, e.g. by applying the principal direction algorithm with multiple filter sets having different centre frequencies in a manner similar to the 2D algorithm described in section 2.2.3. This has not been implemented in the present work, although preliminary tests have shown promising results.

The squared magnitude sum of the shift estimates is invariant to both principal direction and rotation axis, i.e.

$$\sum_{(i,j)} b^2_{(i,j)} = C$$  \hfill (6.29)

where $b_{(i,j)}$ denotes the magnitude of the centre of gravity vector $b^{(i,j)}$, and $C$ is a constant depending on the curvature $\kappa_1$, the radial frequency function of the filters and the certainty of the orientation estimates.

### 6.5 Improvements of the Principal Direction Algorithm

Experiments have shown that the performance of the algorithm is good also in relatively noisy neighbourhoods. This is partly because of the high quality of the orientation estimates and partly because of the vector sum strategy where incompatible events oppose each other. But curvature estimation is, almost by definition, noise sensitive and this section presents methods to improve the accuracy of the estimates.

#### 6.5.1 Averaging

The use of a continuous data representation makes it possible to use filtering on the principal direction estimates. Consistency examination of the principal directions within the local neighbourhood is done by a simple averaging.

$$b'_{(i,j)} = b_{(i,j)} * h_a$$  \hfill (6.30)

where $h_a$ is the averaging filter and $*$ denotes convolution.

It is of interest to study the behaviour of the averaging operation with respect to the new values of $b_{(i,j)}$ (i.e. $P$) and $B$ (eq. (6.26)) when applied to different types of neighbourhoods.

- $k_2 \approx 0$, m constant and different $t_1$. This corresponds to a parabolic neighbourhood. A 'normal' average of $t_1$ will be computed. The relationship between the eigenvalues $\lambda_1'$ and $\lambda_2'$ of the matrix $B$ is not affected.

95
• \( \kappa_2 \approx 0 \), different \( m \) and constant \( t_1 \). A proper average of \( m \) will be computed. The relationship between the eigenvalues \( \lambda'_1 \) and \( \lambda'_2 \) of the matrix \( B \) is not affected.

• \( \kappa_2 \approx 0 \), different \( m \) and different \( t_1 \). The average of the individual \( b_{(i,j)} \) are weighted with the corresponding \( m_k \) (\( i \neq j \neq k \neq i \)). The ratio \( \lambda'_2 / \lambda'_1 \) increases and the eigenvector \( e'_1 \) of the matrix \( B \) is the new estimate of \( t_1 \).

This list can be made longer, but these few examples give some insight in the effect of averaging.

6.5.2 Orientation Consistency

One can take into account that the principal directions and the normal vector of the surface are perpendicular. This relation gives

\[
\mathbf{T}b_{(i,j)} = 0
\]  
(6.31)

Observe that this is valid for both surfaces and curves. The value of \( b_{(i,j)} \) should be updated if eq. (6.31) is not fulfilled. Two different update schemes are now proposed. The orientation tensor can be written as

\[
\mathbf{T} = \lambda_1 \hat{e}_1 \hat{e}_1^T + \lambda_2 \hat{e}_2 \hat{e}_2^T + \lambda_3 \hat{e}_3 \hat{e}_3^T
\]  
(6.32)

Surface neighbourhoods will have \( \lambda_2 \approx \lambda_3 \approx 0 \) and curve neighbourhoods will have \( \lambda_3 \approx 0 \). Assume this type of neighbourhood:

\[
\mathbf{T} = \lambda_1 \hat{e}_1 \hat{e}_1^T + \lambda_2 \hat{e}_2 \hat{e}_2^T
\]  
(6.33)

Insert eq. (6.33) into eq. (6.31)

\[
\mathbf{T}b_{(i,j)} = \lambda_1 \hat{e}_1 \hat{e}_1^T b_{(i,j)} + \lambda_2 \hat{e}_2 \hat{e}_2^T b_{(i,j)} = \lambda_1 (\hat{e}_1 \cdot b_{(i,j)}) \hat{e}_1 + \lambda_2 (\hat{e}_2 \cdot b_{(i,j)}) \hat{e}_2
\]  
(6.34)

This implies that the result will be a vector pointing according to the two eigenvectors. The vector will have the length

\[
b_{(i,j)} \sqrt{\lambda_1^2 \cos^2 \phi_1 + \lambda_2^2 \cos^2 \phi_2}
\]  
(6.35)

where \( b_{(i,j)} \) is the magnitude of \( b_{(i,j)} \) and \( \phi_i \) denotes the angle between \( b_{(i,j)} \) and \( e_i \). Remember that the sum of the squared orientation tensor elements equals the sum of the squared eigenvalues, i.e.

\[
\sqrt{\sum t_{ij}^2} = \sqrt{\sum \lambda_i^2}
\]  
(6.36)

Using this normalization and a unit length \( b_{(i,j)} \), i.e. \( \hat{b}_{(i,j)} \) gives

\[
\frac{\sqrt{\lambda_1^2 \cos^2 \phi_1 + \lambda_2^2 \cos^2 \phi_2}}{\sqrt{\lambda_1^2 + \lambda_2^2}} = \frac{\| \mathbf{T} \hat{b}_{(i,j)} \|}{\sqrt{\sum t_{ii}^2}}
\]  
(6.37)
This expression can be used to formulate a strategy to decrease the magnitude of \( b_{(i,j)} \). This is done by

\[
b'_{(i,j)} = (1 - \frac{\|Tb_{(i,j)}\|}{\sqrt{\sum t_{ij}^2}}) b_{(i,j)}
\]

(6.38)

This is a rather simple magnitude reduction. It is more sophisticated also to modify the direction of \( b_{(i,j)} \) using

\[
b'_{(i,j)} = b_{(i,j)} - \frac{Tb_{(i,j)}}{\sqrt{\sum t_{ij}^2}}
\]

(6.39)

This results in \( b_{(i,j)} \)-vectors which fulfil eq. (6.31).

### 6.5.3 Eigenvalue Consistency

Both the eigenvalues \( \lambda_i \) of the orientation tensor \( T \) and the eigenvalues \( \lambda'_i \) the matrix \( B \) (eq. (6.26)) can be utilized. The eigenvalues of \( T \) discriminate between surfaces and curves and describe how well defined the orientation of the local neighbourhood is. A list of the different cases can be found at the beginning of chapter 5.2. The eigenvalues of \( B \) discriminate between local neighbourhoods of one and two principal directions of curvature. The smallest eigenvalue \( \lambda'_3 \) should always be close to zero.

\[
b'_{(i,j)} = \frac{\lambda'_1 - \lambda'_2}{\lambda'_1} (e'_1 \cdot b_{(i,j)}) e'_1 + \frac{\lambda'_2 - \lambda'_3}{\lambda'_2} (e'_2 \cdot b_{(i,j)}) e'_2
\]

(6.40)

Other ways of combining the information supplied by the eigenvalues are of course possible, e.g. decrease the magnitude of the \( b_{(i,j)} \) vectors when the orientation estimates are uncertain.

### 6.6 Local Shape Analysis

It is necessary to include scale analysis to obtain estimates of \( \kappa_1 \) and \( \kappa_2 \), e.g. apply the principal direction algorithm repetitively with different centre frequencies of the filters. Such a scheme is used in the 2D implementation. But there is a number of local shape features which can be obtained without scale analysis.

#### 6.6.1 Umbilics

It is of interest to estimate the relationship between \( \kappa_1 \) and \( \kappa_2 \). Neighbourhoods with \( \kappa_1 = \kappa_2 \) are umbilics (chapter 6.1). Recall that eq. (6.28) gives access to the principal curvatures scaled by an unknown factor \( X \). Comparison of the two estimates will eliminate \( X \). One such comparison formula is

\[
\left( \frac{2k_1k_2}{k_1^2 + k_2^2} \right)^\alpha
\]

(6.41)

where \( \alpha \) is a parameter used to control the characteristics of the estimate. This estimate does not require explicit solution of the eigenvalues of eq. (6.26) since

\[
\sum b_{(i,j)}^2 = k_1^2 + k_2^2 \sim \kappa_1^2 + \kappa_2^2
\]

(6.42)
and
\[ \sum ||b_{(i,j)} \times b_{(k,l)}||^2 = k_1^2k_2^2 \sim k_1^2k_2^2 \] (6.43)
which gives
\[ \frac{2\sqrt{\sum ||b_{(i,j)} \times b_{(k,l)}||^2}}{\sum b_{(i,j)}^2} = \frac{2k_1k_2}{k_1^2 + k_2^2} \] (6.44)

### 6.6.2 Gaussian Curvature

The gaussian curvature \( \kappa_1\kappa_2 \) is another interesting shape feature. This requires estimation of the sign of the principal curvatures as well. Ellipsoid surface patches have positive gaussian curvature, cylinder neighbourhoods have zero gaussian curvature and saddle areas have negative gaussian curvature.

The values of \( \kappa_1 \) and \( \kappa_2 \) are not explicitly known without the scale space extension of the algorithm. It is possible to compute another feature, the Eggvalue. This parameter has the same sign properties as the gaussian curvature and it is computed by

\[ \sum c_{ii} \] (6.45)

i.e. the eggvalue is the trace of a matrix \( C \), where \( C \) is computed as

\[
C = \begin{pmatrix}
  \vdots & \vdots & \vdots \\
  b_{(1,3)} \times b_{(1,2)} & b_{(1,2)} \times b_{(2,3)} & b_{(2,3)} \times b_{(1,3)} \\
  \vdots & \vdots & \vdots 
\end{pmatrix} \] (6.46)

### 6.7 Examples

The \( b_{(1,2)} \) volume obtained from a synthetic grey level volume containing a cylinder is visualized in fig. 6.5 as a 3D vector plot with short vectors excluded. A similar visualization technique is used in fig. 6.6 and 6.7. Fig. 6.6 contains the \( b_{(1,2)} \) vectors describing the principal directions of an ellipsoid. The vectors are in this case directed as a mixture of \( t_1 \) and \( t_2 \). Separation of the principal directions using eq. (6.27) gives the result in fig. 6.7 which contains the parts of the \( b_{(1,2)} \) vectors describing \( t_1 \).

Fig. 6.8 contains three synthetic test volumes. An ellipsoid, a cylinder and a saddle shape. Note that both the cylinder and the saddle shape are truncated at the volume edges. Fig. A.11 shows the result of the local shape analysis. The eggvalue (eq. (6.45)) is displayed in the top row with green colour to indicate positive values and red colour to indicate negative values. The relation between the principal curvature magnitudes (eq. (6.41)) is displayed in the bottom row with green colour for cylinder neighbourhoods, red for umbilic neighbourhoods and blue in between. The certainty of the estimates are in all images represented as image intensity.

Processing of a ‘real-world’ volume is displayed in fig. A.12 and A.13. The head in fig. 5.2 is used as input data. Fig. A.12 displays the result obtained by a straightforward differential geometry approach. The volume is first thresholded and made into a range image. Filters are then used to obtain estimates of the first and second
Figure 6.5: A look inside the $b_{(1,2)}$ description of a cylinder

Figure 6.6: The $b_{(1,2)}$ description of an ellipsoid
Figure 6.7: The $b_{(1,2)}$ description of $t_1$

Figure 6.8: Three different shapes; egg, cylinder and neck
fundamental form and combining them using eq. (6.11) to obtain an estimate of the Gaussian curvature. This produces reasonable estimates of the sign of the Gaussian curvature but the magnitudes of the estimates are extremely unreliable, with the consequence of many erroneous sign estimates when $\kappa_1 \kappa_2 \approx 0$. The estimates in fig. A.12 are therefore produced by combination over two scales. No total estimate is produced if the two estimate disagree (black colour). Green colour indicates positive Gaussian curvature and red colour negative Gaussian curvature.

A more reasonable magnitude estimate is produced by the eggvalue algorithm (fig. A.13), although the magnitude should be interpreted as a certainty measure and not an estimate of $\kappa_1 \kappa_2$. Good results have been obtained on other medical volumes.

6.8 Summary

A tensor representation of principal directions of curvature in 3D and an algorithm which produces this representation have been presented. The principal directions are estimated by a combination of the filter outputs obtained by convolution of the orientation data. One major benefit is that no thresholding is required. Another is that the representation is continuous. The use of the centre of gravity vectors together with the orientation tensor as representation eliminates the problem of handling the problems with discontinuities. The alternative representations of the local frame (the two principal directions + the surface normal) have for instance discontinuity problems in the neighbourhood of umbilic points. A benefit of using a continuous representation is that relaxation can be implemented very simple: by averaging.

Algorithms for detection of umbilic points and estimation of the sign of the Gaussian curvature has also been presented. Results of processing both synthetic and real-world data have been given.

Appendices

6.A  The Continuous Representation of Principal Directions of Curvature

The ‘uniqueness’ requirement described in Knutsson [60] is demonstrated by rotation of P:

Rotating P around t  A rotation of $\pi$ produces the opposite principal direction. A rotation of $2\pi$ produces the same principal direction.

Rotating P around n  A rotation of $\pi$ produces the opposite principal direction. A rotation of $2\pi$ produces the same principal direction.

Rotating P around x  A rotation of $\pi$ produces the same principal direction. Rotation invariance is proved in appendix 6.1 after the introduction of the algorithm.
6.B The Complex-Valued Orientation Representation

The problem of finding a suitable representation for orientation in 3D is discussed in Knutsson [60]. A representation is regarded suitable if it meets three basic requirements: **Uniqueness, Uniformity and Polar separability.**

- **The 'uniqueness' requirement:**
  The mapping should map the gradient vectors \( x \) and \(-x\) onto the same value in the representation space i.e
  \[
  M(x) = M(-x)
  \] (6.47)

- **The 'uniform stretch' requirement:**
  The mapping should locally preserve the angle metric of the original space, i.e a change of the orientation of \( x \) with \( \Delta \) should change the orientation value of the representation of \( x \) with \( S\Delta \) where \( S \) is a 'stretch' constant.

- **The 'polar separability' requirement:**
  As the information carried by the magnitude of the original vector \( x \) does not normally depend on the vector angle, it is reasonable to require that:
  \[
  ||M(x)|| = f(||x||)
  \] (6.48)
  i.e the norm of \( M(x) \) is independent of the direction of \( x \).

The uniqueness criterion is verified by insertion of \( x \) and \(-x\) into eq. (6.16)–(6.18).

The uniform stretch criterion is verified by examination of a transformation of \( x \) by use of a rotation matrix \( Q \). This results in magnitude changes in the individual complex values of the mapping but this is not the issue here (see the polar separability criterion).

Express \( x \) as \( x(\omega_{12}, \omega_{13}, \omega_{23}) \) where \( x \) is the magnitude of the vector \( x \) and \( \omega_{12}, \omega_{13} \) and \( \omega_{23} \) denotes angles, \( \omega_{12} \) is the angle obtained by projection of \( x \) into the \( x_1-x_2 \)-plane. A rotation changes the angles by addition of \( \Delta_{12}, \Delta_{13} \) and \( \Delta_{23} \). The complex numbers representing \( x \) will originally have the arguments \( 2\omega_{12}, 2\omega_{13} \) and \( 2\omega_{23} \) and the rotation changes them by addition of \( 2\Delta_{12}, 2\Delta_{13} \) and \( 2\Delta_{23} \), i.e the stretch constant equals 2.

The polar separability is fulfilled by using the following norm for the representation.

\[
||M(x)|| = \sum_{i=1}^{3} r_i
\] (6.49)

where \( r_1 \) denotes the magnitude of the complex number obtained by eq. (6.16). Computation gives

\[
r_1^2 = \frac{1}{x^2}[(x_1^2 - x_2^2)^2 + 4x_1^2x_2^2] = \frac{1}{x^2}(x_1^2 + x_2^2)^2
\] (6.50)
\[
r_2^2 = \frac{1}{x^2}(x_1^2 + x_3^2)^2
\] (6.51)
\[
r_3^2 = \frac{1}{x^2}(x_2^2 + x_3^2)^2
\] (6.52)

and

\[
||M(x)|| = \frac{2}{x}(x_1^2 + x_2^2 + x_3^2) = 2x
\] (6.53)
6.C Projection of 3D Normal Vectors into 2D Vectors

The 3D vectors represent surface normals. The 2D vectors are in reality (eq. (6.16)-(6.18)) represented with ‘double angle’. This is disregarded in the computations here since it will only affect the speed of the angular changes and not the main direction of angular change (= the principal direction).

Assume a 3D neighbourhood of 3D-vector valued voxels. Let the neighbourhood be of parabolic type, i.e. located on the surface of a cylinder. Elliptic and hyperbolic surface types will be dealt with later. Separate the neighbourhood in one scalar 3D neighbourhood, representing the length of the original 3D-vectors, and one 3D neighbourhood of unit vectors, representing the direction of the vector field. The neighbourhood has a curvature \( \kappa_1 \) in the principal direction \( t_1 \). The curvature rotates the normal vectors with respect to a rotation axis \( n \).

A number of constraints originating from the behaviour of normal vectors on a cylinder can be used on \( n \) and \( t_1 \), but this would result in a less general model. The model describes one-dimensional (and later two-dimensional) direction variations of 3D vectors in arbitrary 3D neighbourhoods. The surface-type neighbourhood constraints may well be used in the later stage of processing.

The 3D neighbourhood containing unit vectors is now expressed in equations. Fig. 6.3 illustrates the model. The centre voxel \((0,0,0)\) has the value (is directed as) \( a \). The other voxels are valued as

\[
\begin{align*}
V_1(\xi) &= a + 2 \cos \frac{\theta}{2} (\sin \frac{\theta}{2} n \times a) + 2 \sin \frac{\theta}{2} n \times (\sin \frac{\theta}{2} n \times a) \\
\theta &= \kappa_1 t_1 \cdot \xi
\end{align*}
\]

(6.54)

\( n \) and \( t \) are unit vectors. \( \xi \) is the coordinate vector.

Eq. (6.54) can be rewritten as

\[
\begin{align*}
v_1(\xi) &= \cos \theta (a_1 - (n \cdot a) n_1) + \sin \theta (n_2 a_3 - n_3 a_2) + (n \cdot a) n_1 \\
v_2(\xi) &= \cos \theta (a_2 - (n \cdot a) n_2) + \sin \theta (n_3 a_1 - n_1 a_3) + (n \cdot a) n_2 \\
v_3(\xi) &= \cos \theta (a_3 - (n \cdot a) n_3) + \sin \theta (n_1 a_2 - n_2 a_1) + (n \cdot a) n_3
\end{align*}
\]

(6.55) (6.56) (6.57)

where \( v(\xi) = (v_1, v_2, v_3) \) is the value of the voxel located at \( \xi \). Note that

\[
n \cdot a = 0
\]

(6.58)

If this does not hold, \( a \) can be rewritten as \( a' = a + d \), where \( a' \perp n \) and \( d \parallel n \). This means that the neighbourhood can be expressed as the sum of two neighbourhoods, one with rotating vectors and one with a constant vector field \( (d) \). An input of a constant vector field neighbourhood to the algorithm produces a zero output and the constant field can be disregarded.

This simplifies the equations to

\[
\begin{align*}
v_1 &= \cos \theta a_1 + \sin \theta (n_2 a_3 - n_3 a_2) \\
v_2 &= \cos \theta a_2 + \sin \theta (n_3 a_1 - n_1 a_3) \\
v_3 &= \cos \theta a_3 + \sin \theta (n_1 a_2 - n_2 a_1)
\end{align*}
\]

(6.59)

Introduce for simplicity a new vector \( c = n \times a \) and rewrite eq. (6.59).

\[
\begin{align*}
v_1 &= a_1 \cos \theta + c_1 \sin \theta \\
v_2 &= a_2 \cos \theta + c_2 \sin \theta \\
v_3 &= a_3 \cos \theta + c_3 \sin \theta
\end{align*}
\]

(6.60)
Three different 3D to 2D projections can be made. A closer look at $v_1 + iv_2$ yields

$$
\begin{pmatrix}
  v_1 \\
  v_2
\end{pmatrix} = \begin{pmatrix}
  a_1 \cos \theta + c_1 \sin \theta \\
  a_2 \cos \theta + c_2 \sin \theta
\end{pmatrix} = \begin{pmatrix}
  \frac{a_1 + c_2}{2} \cos \theta - \frac{a_2 - c_1}{2} \sin \theta \\
  \frac{a_1 + c_2}{2} \sin \theta + \frac{a_2 - c_1}{2} \cos \theta
\end{pmatrix} + \begin{pmatrix}
  \frac{a_1 - c_2}{2} \cos \theta + \frac{c_1 + a_2}{2} \sin \theta \\
  \frac{a_1 - c_2}{2} \sin \theta - \frac{c_1 + a_2}{2} \cos \theta
\end{pmatrix}
$$

Combining this with the scalar 3D neighbourhood representing the length of the original 3D-vectors gives the following expression for the 3D neighbourhood of complex valued numbers.

$$
f_1(\xi) = r(\xi) \left( \sqrt{\frac{(a_1 + c_2)^2}{2} + \frac{(a_2 - c_1)^2}{2}} \exp(i\theta(\xi)) \right)$$

$$
+ \sqrt{\left( \frac{a_1 - c_2}{2} \right)^2 + \left( \frac{a_2 + c_1}{2} \right)^2 \exp(-i\theta(\xi)) + X}
$$

where $r(\xi)$ denotes the 3D-vector length and $X$ denotes the contribution from the constant vector field $d$ introduced in eq. (6.58).

Taking this expression to the Fourier domain gives

$$
F_1(u) = \sqrt{\left( \frac{a_1 + c_2}{2} \right)^2 + \left( \frac{a_2 - c_1}{2} \right)^2} R(u - \kappa t)$$

$$
+ \sqrt{\left( \frac{a_1 - c_2}{2} \right)^2 + \left( \frac{a_2 + c_1}{2} \right)^2} R(u + \kappa t) + X R(u)
$$

The projection of the 3D vector into a complex number thus results into not one but two shifts in the Fourier domain of the local neighbourhood. The two shifts are opposite directed and the centre of gravity of the Fourier spectra will be located somewhere on the line defined by $t$, either on the ‘positive’ or the ‘negative’ side depending on magnitude of the two shifts.

This is sufficient for proving that the subspaces yields the correct shift for parabolic neighbourhoods.

### 6.D Generalization of the Parabolic Model to Elliptic and Hyperbolic Neighbourhoods

The algorithm must, however, be able to keep track of two principal directions, $t_1$ and $t_2$, to be able to handle arbitrary surfaces. Recall from eq. (6.19) that the approximation of cylinder type neighbourhoods is $\theta = \kappa t \cdot \xi$. The more general model with two principal directions should be consistent with this approximation and to Euler’s theorem (eq. (6.10)). The neighbourhood can be decomposed into 2D slices which intersect the 2D-plane spanned by the two principal directions of curvature at the angle $\alpha = t_1 + it_2$. The rotation of the 3D vectors (and the complex-valued mappings) becomes more and more one-dimensional when the slice thickness decreases. The direction of the rotation changes will coincide with either the direction defined by $\alpha$ or the opposite direction $\alpha + \pi$.

A curvature description for this particular slice should describe the amount of curvature, $\kappa$ and the direction.
The summation of the slice-curvature descriptions gives the total curvature description for the neighbourhood. Euler's theorem can be used to compute the contribution in the $\hat{t}_1$ direction. The contribution for a fixed $\alpha$ is $\kappa \cos \alpha$. Integrating over $\alpha$

$$\int_0^{\pi/2} \cos \alpha (\kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha) d\alpha = \kappa_1$$ (6.64)

It is, due to symmetry, sufficient to integrate in the interval $[0 \ldots \pi/2]$.

Likewise, for $\hat{t}_2$

$$\int_0^{\pi/2} \sin \alpha (\kappa_1 \cos^2 \alpha + \kappa_2 \sin^2 \alpha) d\alpha = \kappa_2$$ (6.65)

This implies that the direction of angular change for a complex-valued representation of 3D orientation will equal

$$n_i \kappa_1 \hat{t}_1 + m_i \kappa_2 \hat{t}_2$$ (6.66)

where $i$ is the index of the coordinate not present in the 3D-vector to complex-value mapping.

### 6.E Estimation of Centre of Gravity

The objective is to show that quasi-quadrature filters (eq. (6.22)) are suitable for the task of localizing the centre of gravity of the Fourier spectra. Assume that the Fourier spectra contain one single impulse located at $\rho (\cos \alpha, \cos \beta, \cos \gamma)$, where $\rho$ is the distance from the origin and $\cos \alpha$, $\cos \beta$ and $\cos \gamma$ are the direction cosines.

The filter responses are, for a filter set distributed according to eq. (6.78), equal to

$$q_1 = H_\rho (\rho) / 2 (1 + \cos \alpha)$$ (6.67)
$$q_2 = H_\rho (\rho) / 2 (1 + \cos \beta)$$ (6.68)
$$q_3 = H_\rho (\rho) / 2 (1 + \cos \gamma)$$ (6.69)
$$q_4 = H_\rho (\rho) / 2 (1 - \cos \alpha)$$ (6.70)
$$q_5 = H_\rho (\rho) / 2 (1 - \cos \beta)$$ (6.71)
$$q_6 = H_\rho (\rho) / 2 (1 - \cos \gamma)$$ (6.72)

Evaluate eq. (6.21)

$$\sum q_k \hat{n} = (q_1 - q_4, q_2 - q_5, q_3 - q_6) = H_\rho (\rho)(\cos \alpha, \cos \beta, \cos \gamma)$$ (6.73)

### 6.F Obtaining Six Complex Filter Responses by Four Scalar Filters

The objective is to show that eq. (6.21) for quasi-quadrature filters distributed according to eq. (6.78) can be computed with four scalar filters.

Examination of e.g $H_1(u)$ gives

$$H_1(u) = H_\rho (u) \cdot H_\theta = \frac{H_\rho (u)}{2} + \frac{H_\rho \cos \alpha}{2}$$ (6.74)
Note that $H_p(u)$ is an even function and that $H_p(u) \cos \alpha$ is an odd function. This implies that the spatial representation of $H_1$ is Hermitian (Bracewell [27]), i.e.

$$h_1(\xi) = e_1(\xi) + i o_1(\xi)$$

(6.75)

where $e_1$ is the even part of $h_1$ and $o_1$ is the odd part.

Observe that

$$e_1(\xi) = \frac{h_p(\xi)}{2} = e_2(\xi) = \ldots = e_8(\xi).$$

(6.76)

That is, the real part of the complex valued filters $h_i$ are all identical (and of Laplace-type). It is thus sufficient to convolve with one filter to obtain the response from the real part of the eight filters.

The second observation that can be made is

$$e_1(\xi) = -e_4(\xi)$$

$$e_2(\xi) = -e_5(\xi)$$

$$e_3(\xi) = -e_6(\xi)$$

(6.77)

This implies that the convolution with the imaginary part of the six filters can be obtained by four convolutions and sign changes.

It is consequently enough to perform four $(1 + 3)$ scalar convolutions to obtain six complex convolutions.

### 6.G Fourier Domain Directions of the Filter Set

The 6 normal vectors are given by:

$$\hat{n}_1 = (1 \ 0 \ 0)^T$$

$$\hat{n}_2 = (0 \ 1 \ 0)^T$$

$$\hat{n}_3 = (0 \ 0 \ 1)^T$$

$$\hat{n}_4 = (-1 \ 0 \ 0)^T$$

$$\hat{n}_5 = (0 \ -1 \ 0)^T$$

$$\hat{n}_6 = (0 \ 0 \ -1)^T$$

(6.78)

### 6.H Implementation of the Algorithm

1. Convolvve the orientation data (six scalars) with the six complex-valued filters, i.e. perform $6 \times 6 \times 2$ scalar convolutions.

2. Compute for each of the index combinations $i, j$

$$b_{(i,j)} = \sum_{k=1}^{6} q_{ijk} \hat{n}_k$$

(6.79)

where $n_k$ denotes the Fourier domain direction of filter $k$. The scalar value $q_{ijk}$ is computed by

$$q_{ijk} = \sqrt{q_{ijk}^2 + q_{ijko}^2}$$

(6.80)

with

$$q_{ijk} = h_{ke} * t_{ii} - h_{ke} * t_{jj} - 2h_{ko} * t_{ij}$$

(6.81)

$$q_{ijko} = 2h_{ke} * t_{ij} + h_{ko} * t_{ii} - h_{ko} * t_{jj}$$

(6.82)

where $*$ denotes convolution, $h = h_{ke} + ih_{ko}$ denotes the spatial filter function of filter $k$ and $t_{ii}, t_{ij}$ and $t_{jj}$ are elements of the orientation tensor.
There are three different index combinations; \( (1, 2), (1, 3) \) and \( (2, 3) \) and eq. (6.79) will consequently produce three different \( b \)-vectors; \( b_{(1,2)}, b_{(1,3)} \) and \( b_{(2,3)} \).

### 6.1 Rotation Invariance

This criterion can be expressed as

\[
b^2_{(1,2)} + b^2_{(1,3)} + b^2_{(2,3)} = X^2
\]

where \( b_{(i,j)} \) denotes the magnitude of one of the centre of gravity estimates. \( X \) only depends on the amount of curvature, the certainty of the orientation estimates and the frequency characteristics of the quasi-quadrature filter used. That is, \( X \) is invariant to the principal directions. This will now be demonstrated for parabolic neighbourhoods with one principal direction \( t_1 \).

Returning to eq. (6.63) and its Fourier shifts and combining it with eq. (6.79) gives that the 'positive' shift will result in a vector pointing in the \( t_1 \) direction with the magnitude

\[
Y \cdot \sqrt{\left( \frac{a_1 + c_2}{2} \right)^2 + \left( \frac{a_2 - c_1}{2} \right)^2}
\]

The 'negative' shift will likewise produce a vector pointing in the \(-t_1\) direction with magnitude

\[
Y \cdot \sqrt{\left( \frac{a_1 - c_2}{2} \right)^2 + \left( \frac{a_2 + c_1}{2} \right)^2}
\]

where \( Y \) depends on the frequency characteristics of the quasi-quadrature filter together with the shape of \( R(u) \), the local Fourier transform of the magnitude function \( r(\xi) \) (eq. (6.62)).

The summation of the two opposite directed vectors of eq. (6.63) gives \( b_{(1,2)} \) of eq. (6.79) with a magnitude equal to

\[
b_{(1,2)} = \frac{1}{2} Y \sqrt{a_1^2 + c_2^2 + 2a_1c_2 + a_2^2 + c_1^2 - 2a_2c_1 - \frac{1}{2} Y \sqrt{a_1^2 + c_2^2 - 2a_1c_2 + a_2^2 + c_1^2 + 2a_2c_1}}
\]

and consequently

\[
b^2_{(1,2)} = Y^2/2(a_1^2 + c_2^2 + c_1^2 + a_2^2) - Y^2/2\sqrt{(a_1^2 + c_2^2 + c_1^2 + a_2^2)^2} - (2a_1c_2 - 2c_1a_2)^2
\]

Likewise

\[
b^2_{(1,3)} = Y^2/2(a_1^2 + c_3^2 + c_1^2 + a_3^2) - Y^2/2\sqrt{(a_1^2 + c_3^2 + c_1^2 + a_3^2)^2} - (2a_1c_3 - 2c_1a_3)^2
\]

\[
b^2_{(2,3)} = Y^2/2(a_2^2 + c_3^2 + c_2^2 + a_3^2) - Y^2/2\sqrt{(a_2^2 + c_3^2 + c_2^2 + a_3^2)^2} - (2a_2c_3 - 2c_2a_3)^2
\]

Observe that

\[
(a_1c_2 - a_2c_1)^2 = m_3^2
\]

\[
(a_1c_3 - a_3c_1)^2 = m_2^2
\]

\[
(a_2c_3 - a_3c_2)^2 = m_1^2
\]

107
and since $a, c$ and $m$ are unit vectors, 
\begin{align}
a_1^2 + a_2^2 + a_3^2 &= 1 \\
c_1^2 + c_2^2 + c_3^2 &= 1 \\
m_1^2 + m_2^2 + m_3^2 &= 1 
\end{align}

and orthogonal 
\begin{align}
a_1 c_1 + a_2 c_2 + a_3 c_3 &= 0 \\
a_1 m_1 + a_2 m_2 + a_3 m_3 &= 0 \\
c_1 m_1 + c_2 m_2 + c_3 m_3 &= 0. 
\end{align}

Finally, observe that for orthogonal unit vectors $m, a, c,$ 
\begin{align}
m_3^2 + a_3^2 + c_3^2 &= 1 
\end{align}

since the expression is true for the triplet $(100)^T, (010)^T, (001)^T$ and rotating the triplet with an arbitrary rotation matrix 
\begin{align}
\begin{pmatrix}
\cos \gamma \cos \alpha - \cos \beta \sin \alpha \sin \gamma & \cos \gamma \sin \alpha + \cos \beta \cos \alpha \sin \gamma & \sin \gamma \sin \beta \\
- \sin \gamma \cos \alpha - \cos \beta \sin \alpha \cos \gamma & - \sin \gamma \sin \alpha + \cos \beta \cos \alpha \cos \gamma & \cos \gamma \sin \beta \\
\sin \beta \sin \alpha & - \sin \beta \cos \alpha & \cos \beta
\end{pmatrix}
\end{align}
gives 
\begin{align}
m_3 &= \sin \beta \sin \alpha \\
a_3 &= - \sin \beta \cos \alpha \\
c_3 &= \cos \beta
\end{align}

from which follows 
\begin{align}
m_3^2 + a_3^2 + c_3^2 &= 1 
\end{align}

Insertion of eq. (6.90) and eq. (6.91) into eq. (6.87) gives 
\begin{align}
b_{1,2}^2 = Y^2 - (a_3 + c_3) - Y^2/2 \sqrt{(2m_1^2 + 2m_2^2 + 2m_3^2 - a_3 - c_3)^2 - 4m_3^2} 
\end{align}

The expression inside the square root requires a further investigation. 
\begin{align}
(2m_1^2 + 2m_2^2 + 2m_3^2 - a_3 - c_3)^2 = \\
= (2m_1^2 + 2m_2^2 - a_3 - c_3)^2 + 4m_3^2 + 8m_1^2 m_3^2 + 8m_2^2 m_3^2 + 4m_3^2 a_3^2 + 4m_3^2 c_3^2 - 4m_2^2 = \\
= (2m_1^2 + 2m_2^2 - a_3 - c_3)^2 + 4m_3^2 (m_3^2 + m_2^2 + m_1^2)^2 = 4m_3^2 (m_1^2 + m_2^2 - a_3 - c_3)^2
\end{align}

Finally using eq. (6.93) in eq. (6.100) gives 
\begin{align}
(2m_1^2 + 2m_2^2 - a_3 - c_3)^2
\end{align}

for the expression inside the square root, which simplifies eq. (6.97) to 
\begin{align}
b_{1,2}^2 = Y^2 - Y^2/2 (a_3 + c_3) - Y^2/2 (2m_1^2 + 2m_2^2 - a_3 - c_3) = Y^2 m_3^2 
\end{align}

Likewise 
\begin{align}
b_{1,3}^2 &= Y^2 m_2^2 \\
b_{2,3}^2 &= Y^2 m_1^2
\end{align}

and 
\begin{align}
b_{1,2}^2 + b_{1,3}^2 + b_{2,3}^2 = Y^2
\end{align}

This implies rotation invariance.
6.J Proof of eq. (6.44)

Examine a point on a surface with principal directions \( t_1 \) and \( t_2 \) of curvatures \( \kappa_1 \) and \( \kappa_2 \). The \( b_{(i,j)} \) estimates are

\[
\begin{align*}
    b_{(1,2)} &= k_1 m_3 t_1 + k_2 m_2 t_2 \\
    b_{(1,3)} &= k_1 m_2 t_1 + k_2 m_3 t_2 \\
    b_{(2,3)} &= k_1 m_1 t_1 + k_2 m_1 t_2
\end{align*}
\]

(6.106)

where \( m \) is the rotation axis of \( t_1 \) and \( m' \) is the rotation axis of \( t_2 \). Computing the magnitude of the \( b_{(i,j)} \)-vectors gives

\[
\begin{align*}
    b_{(1,2)} &= \sqrt{k_1^2 m_3^2 + k_2^2 m_2^2} \\
    b_{(1,3)} &= \sqrt{k_1^2 m_2^2 + k_2^2 m_3^2} \\
    b_{(2,3)} &= \sqrt{k_1^2 m_1^2 + k_2^2 m_1^2}
\end{align*}
\]

(6.107)

since \( t_1 \) and \( t_2 \) are perpendicular. Using eq. (6.107) to compute the denominator of eq. (6.44) verifies

\[
\sum b_{(i,j)}^2 = k_1^2 + k_2^2
\]

(6.108)

since \( m \) and \( m' \) are unit vectors.

Using eq. (6.106) to compute the cross products gives

\[
\begin{align*}
    b_{(1,2)} \times b_{(1,3)} &= k_1 k_2 (m_3 m_4 - m_2 m_3) t_1 \times t_2 \\
    b_{(1,2)} \times b_{(2,3)} &= k_1 k_2 (m_3 m_4 - m_2 m_3) t_1 \times t_2 \\
    b_{(1,3)} \times b_{(2,3)} &= k_1 k_2 (m_2 m_4 - m_3 m_3) t_1 \times t_2
\end{align*}
\]

(6.109)

Continuing with summation of the squared magnitudes of the cross products

\[
\sum \| b_{(i,j)} \times b_{(k,l)} \|^2 = k_1^2 k_2^2 \left( m_2^2 m_2^2 + m_3^2 m_3^2 - 2m_2 m_3 m_3 m_3 + m_3^2 m_2^2 + m_3^2 m_3^2 - 2m_1 m_3 m_4 m_4 + m_3^2 m_1^2 + m_3^2 m_4^2 - 2m_1 m_2 m_4^2 \right)
\]

(6.110)

The rotation axis vectors \( m \) and \( m' \) are perpendicular, i.e. \( m \cdot m' = 0 \). Squaring this relation gives

\[
(m \cdot m')^2 = m_1^2 m_1^2 + m_2^2 m_2^2 + m_3^2 m_3^2 + 2m_2 m_3 m_4 m_4 + 2m_1 m_3 m_4 m_4 + 2m_1 m_2 m_4 m_4 = 0
\]

(6.111)

which, inserted into eq. (6.110) gives

\[
\sum \| b_{(i,j)} \times b_{(k,l)} \|^2 = k_1^2 k_2^2 (m_1^2 + m_2^2 + m_3^2) (m_1^2 + m_2^2 + m_3^2) = k_1^2 k_2^2 \sim k_1^2 k_2^2
\]

(6.112)

This verifies the nominator of eq. (6.44).

6.K Proof of eq. (6.45)

Denoting \( t_1 \) with \( t \) and \( t_2 \) with \( t' \) for simpler indices gives

\[
\sum c_{ii} = k_1 k_2 (m_3 m_2 - m_3 m_2) (t_3 t_3' - t_3 t_3') + k_1 k_2 (m_3 m_1 - m_3 m_1) (t_3 t_3' - t_1 t_3') + k_1 k_2 (m_2 m_1 - m_2 m_1) (t_3 t_3' - t_2 t_3')
\]

(6.113)
It is easy to verify (e.g., by rotating a parabolic surface patch around the surface normal) that for an elliptic surface

\[
\begin{align*}
    \begin{cases}
        m' &= t \\
        t' &= -m
    \end{cases}
\end{align*}
\] (6.114)

Insertion into eq. (6.113) gives

\[
\sum c_{ii} = k_1 k_2 [(t_3 m_2 - t_2 m_3)^2 + (t_1 m_3 - t_3 m_1)^2 + (t_2 m_1 - t_1 m_2)^2] = k_1 k_2
\] (6.115)

Similarly for a hyperbolic surface

\[
\begin{align*}
    \begin{cases}
        m' &= t \\
        t' &= m
    \end{cases}
\end{align*}
\] (6.116)

which gives

\[
\sum c_{ii} = -k_1 k_2
\] (6.117)
Chapter 7

Using the Principal Direction Algorithm for Time Sequence Processing

The interest in time sequence analysis have increased during the last decade. One reason is of course the introduction of more powerful computers. Another is the strong indications that the Active Vision approach [3, 6, 8] may solve many previously infeasible computer vision problems. Reliable optical flow algorithms are some of the components in this methodology.

This chapter describes how the algorithms for estimation of orientation and principal directions, described in chapter 5 and 6, can be used for time sequence processing, where the orientation and curvature features represent entities such as velocity and acceleration.

7.1 Introduction

Three different approaches can be distinguished in the field of image analysis of time sequences.

- **Matching** 2D image processing techniques are used on one frame at a time to extract feature points, curves etc. Matching of the feature extraction result is used to find the corresponding parts in the neighbour frames [49, 50].

- **Derivatives** The solution to the optical flow equation is approximated by the use of gradient filters. The filters are 3D-filters (two spatial dimensions + time) [23, 55].

- **Signal Processing** The 3D data set is analyzed with tools such as the Fourier Transform to design algorithms for the estimation of the planes in the spatio-temporal data set originating from moving lines and curves originating from moving points [38, 1, 47].

The borderlines between these three methods are not always clear, and it is possible to use algorithms which incorporate aspects of all of the three approaches. This chapter will not elaborate on the virtues and drawbacks of the different approaches.
The methods presented here can be considered to belong to the signal processing approach.

A review of biological image motion processing can be found in Nakayama [75]. Recent developments in modelization of the human visual system suggest the existence of frequency and orientation channels representing the local spatial, as well as the local spatio-temporal image spectra, [51, 2]. In the latter case the energy concentration in the local spectra to a particular orientation and frequency channel means a velocity vector with a particular direction and coarseness range. Adelson and Bergen [1] as well as Watson and Ahumada [97] and others have proposed human visual motion sensing models based on the local spectra by means of the spatio-temporal filter responses using separable filters of Gabor type.

The method presented here uses spatio-temporal filtering in a hierarchical structure. Two algorithms, one for estimation of velocity/optical flow and one for estimation of acceleration, are described. The two algorithms are closely related and give a unified hierarchical approach to time sequence processing. Fig. 7.1 shows the algorithm components. Results on both synthetic and real-world data are given. The basic operation is spatio-temporal filtering with polar separable 3D filters (two spatial dimensions and time).

An implication of time being the third dimension is that only half of the local neighbourhood data is available at computation time in a real-time application. The solution to this problem is to use recursive filters (e.g., Dudgeon and Mersereau [37]). However, the design and implementation of recursive quadrature and quasi-quadrature filters are not discussed here.
7.2 Velocity Estimation

It is shown in [60] that the eigenvector corresponding to the largest eigenvalue of the tensor is the normal vector of the plane best describing the neighbourhood. In the case of time sequences a plane means a moving line and the projection of the eigenvector onto the image plane will give the optical flow. In all optical flow algorithms which relies on local operators, the so called aperture problem will give rise to an unspecified velocity component, the component moving along the line. On the other hand the aperture problem does not exist when points are moved in the sequence.

By examining the relations between the eigenvalues it is possible to divide the optical flow estimation into different categories, and use different strategies in the different cases. Case number two above gives a correct estimation of the velocity in the image plane and is thus very important in the understanding of the motion.

The discrimination of the different cases is done by using the following discriminant functions:

\[ p_{\text{plane}} = \frac{\lambda_1 - \lambda_2}{\lambda_1} \]  
(7.1)

\[ p_{\text{line}} = \frac{\lambda_2 - \lambda_3}{\lambda_1} \]  
(7.2)

\[ p_{\text{iso}} = \frac{\lambda_3}{\lambda_1} \]  
(7.3)

These expressions can also be seen as the probability of each case. The largest value decides what case it is.

A beneficial property of the representation is that it easy to compute results from different spatio-temporal scales [98]).

Estimation of velocity in an image sequence is equivalent to the 3D-orientation estimation in the image-time volume. Velocity estimates can be directly obtained from the orientation tensor. Different projections of the orientation tensor provides with information about both spatial orientation of the local structure and the optical flow. Velocity is measured by calculating the energy relation between the spatial dimensions and the time dimension. In areas of moving point type a true velocity estimate is produced. In single orientation areas the velocity component perpendicular to the local structure is produced. The relations between the eigenvalues indicates which type of local neighbourhood that is described. A more detailed description of the algorithm can be found in [14] and Haglund [41].

7.3 Acceleration Estimation

Acceleration in the image often refers to a significant event in the image. It is an important way of finding interesting regions. The addition of acceleration information gives a number of advantages, both as a feature in itself and as an information support to the velocity algorithm.

- The measurement of acceleration indicates weather a linear movement model is sufficient or when an acceleration model has to be incorporated.
• The estimation of acceleration improves the capability to track and discriminate moving objects.

• The discrimination between curvatures originating from accelerating lines, non-accelerating curves and noisy neighbourhoods aids the velocity algorithms in the choice between moving line and moving point.

The spatio-temporal surface/curve supplies information regarding these different cases. An acceleration gives rise to a curvature of the spatio-temporal surface or a curvature/torsion of the spatio-temporal curve. Curvature can also originate from other sources than acceleration. Consider, for example, a circle moving with constant velocity. This indicates that curvature and acceleration are not proportional. A fixed acceleration will give a substantial spatio-temporal curvature for velocities in the range near zero and almost no curvature near the maximum detectable velocity. It is on the other hand not important to have exact estimates of the acceleration for the cases when it only has minor influence on the movements of the object. An acceleration algorithm based on the estimation of spatio-temporal curvature is consequently feasible.

The local curvature of a surface is implicit in the local orientation representation. A plane surface has only one eigenvalue \( \lambda_1 \neq 0 \), but this is not the case for surfaces with curvature. A cylinder type surface could, for instance, have \( \lambda_2 = 0.5 \lambda_1 \). But this could also be the case for a plane surface in a noisy neighbourhood. A curvature description of the local neighbourhood would, however, discriminate between the two cases.

The acceleration algorithm is shortly described by the following steps:

1. Estimate the principal directions of curvature of the spatio-temporal surface.
2. Combine the principal direction estimates with the eigenvector of the orientation tensor corresponding to the optical flow to extract the curvature originating from acceleration.
3. Combine curvature direction estimates from different spatio-temporal channels to obtain an estimate of the acceleration.

7.3.1 From Principal Directions to Acceleration Direction

The eigenvalues and eigenvectors of the orientation tensor are used to choose the direction of curvature \( \mathbf{t} \) originating from acceleration. The correct \( \mathbf{t} \) is pointing in the same direction as the 3D velocity vector, \( \mathbf{v} \). The curvature originating from acceleration is obtained by

\[
\mathbf{b}'_{(i,3)} = (\hat{\mathbf{v}} \cdot \mathbf{b}_{(i,3)})\hat{\mathbf{v}}
\]

A 2D vector pointing in the acceleration direction is obtained by combining the time component of the \( \mathbf{b}' \) vectors in the following manner.

**CASE point**

\[
\mathbf{b}'_{(1,3)3} + i\mathbf{b}'_{(2,3)3}
\]
CASE line/curve translation

\[ b'(1,3)3 + ib'(2,3)3 \]  

CASE line/curve rotation

\[ -b'(1,3)3 - ib'(2,3)3 \]  

The discrimination between point/line is done with the earlier described analysis of the orientation tensor (eq. (7.1)-(7.3)) and \( b'(1,2) \) discriminates between rotation and translation since \( m3 \) (see eq. (6.25)) by necessity equals zero for curvature originating from acceleration originating from translation. Other types of consistency checks, e.g. to ensure that the components of the acceleration direction vector always have the correct sign, are under implementation. Rotating and translating lines, arbitrary movements of points and translation of curves are currently handled correctly by the algorithm. Events such as moving corners etc. are under investigation.

7.3.2 Estimation of amount of acceleration

The spatio-temporal channels \([98]\) supplys the means for estimation of amount of acceleration. Note that subsampling time will increase the spatio-temporal curvature around \( v = 0 \), subsampling the spatial dimensions will increase the spatio-temporal curvature around \( v = v_{\text{max}} \), and subsampling all dimensions increases the spatio-temporal curvature in the whole range of \( v \). This implies that a scale analysis scheme can be used to estimate the amount of acceleration. The mapping from curvature \( \kappa \) to acceleration \( a \) \([90]\) is obtained by

\[ a = \kappa(1 + v^2)^{3/2} \]  

7.4 Results

The moving point case is, as indicated by eq. (7.5), relatively simple. Correct estimates of 3D (two spatial dimensions + time) curves, having both curvature and torsion, are produced by the combination of the information from the orientation tensor \( T \) and the principal direction of curvature tensor \( P \). However, no such examples are included here. The two examples given, one synthetic and one real-world image sequence, have higher degree of difficulty.

The synthetic example demonstrates that curves moving with constant velocity can be discriminated from curvature originating from acceleration. The time sequence contains a circle moving with constant velocity in time-varying white Gaussian noise. The direction of the movement changes once in the sequence. The upper part of fig. A.14 contains one frame in the beginning of the time sequence with the circle moving right and down. The (empty) acceleration estimate is displayed to the right. The lower part contains the frame in which the sudden change of movement occurs. The direction of movement is changed to left and down. This is also indicated in the direction of acceleration estimate to the right where the red colour represents acceleration to the left.

The final example is a time-sequence of a bungy jump taking place at Skipper’s Canyon outside Queenstown, New Zealand. The sequence is recorded with a standard VHS camera, tracking the jumper. The camera zooms during the sequence
with the diaphragm adapting to different light conditions (bright sunshine in combination with the shadow under the bridge). It is hard to deduce the ground truth for all the frames in this sequence\(^1\) but the apparent movement change in the frame chosen in fig. A.15 has good correspondence with the acceleration direction estimate. The bungy chord starts to stretch in this frame and a previously flat fall is changing into a dive. This change, combined with the tracking movements of the camera, gives an acceleration to the left. The result is superimposed on the original frame. The red colour indicates acceleration to the left. This result was obtained by a spatio-temporal subsampling eliminating interlacing artifacts [98] followed by the orientation and principal direction estimation and finally the combination into an estimate of the acceleration direction.

\(^1\)Although I tried to keep my body rigid!
Part III

4D — Volume Sequence Processing

The creatures [Tralfamadorians] were friendly, and they could see in four dimensions. They pitied Earthlings for being able to see only three. They had many wonderful things to teach Earthlings, especially about time.

"SLAUGHTER HOUSE-5"— Kurt Vonnegut
Chapter 8

Estimation of 4D Orientation

Four-dimensional data have been of relatively little interest in the computer vision community\(^1\) but this is rapidly changing. There are at last three different reasons for this:

**Visualization** Methods for the visualization of 4D (or even higher dimensionality) data is evolving within the scientific visualization field (See Nielsen et al [77] for an introduction).

**Imaging** Some techniques for acquisition of time sequences of volumes already exist. The Dynamic Spatial Reconstructor from the Mayo Clinic is fast enough to acquire spatial 3D + time data [85]. A volume-time sequence of a heart beating is available from INRIA-Paris [73].

**Modelling** The use of explicit 4D modeling makes it possible to handle complex vision tasks. The self-driving car of Dickmanns and Graefe [34, 35] represents both 3D space and time in its world model. Another example is scale-space techniques, which when applied on volume data or ordinary time-sequences introduces scale as the fourth dimension.

These facts taken together with the introduction of more powerful computers imply that the need of algorithms for 4D image processing is likely to increase.

Time sequences of volumes are examples of 4D images. Interesting events in this type of data are for instance the orientation and motion of surfaces and lines. These events can all be described in terms of 4D orientation.

This chapter presents the 4D implementation of the orientation algorithm described earlier in this thesis (chapter 2.1 and 5).

8.1 The Tensor Representation

Recall from chapter 5 that the representation of orientation is

\[ T = x^{-1}xx^T \]  \hspace{1cm} (8.1)

\(^1\)4D has been used in other areas, see e.g. [92]
where $x$ is the length of the vector $x$. For $x^T = (x_1, x_2, x_3, x_4)$, $T$ becomes

$$
\frac{1}{x} \begin{pmatrix}
  x_1^2 & x_1x_2 & x_1x_3 & x_1x_4 \\
  x_1x_2 & x_2^2 & x_2x_3 & x_2x_4 \\
  x_1x_3 & x_2x_3 & x_3^2 & x_3x_4 \\
  x_1x_4 & x_2x_4 & x_3x_4 & x_4^2
\end{pmatrix}
$$  (8.2)

The following sections describe the realization of this representation using filtering and tensor summation. Techniques for the visualization of the orientation estimates are also described.

### 8.2 The Algorithm

The 4D implementation uses the same type of quadrature filters used for 2D and 3D orientation estimation (appendix 8.A). The local orientation representation is obtained by linear summation of the quadrature filter output magnitudes

$$
T = \sum_k q_k (\hat{n}_k \hat{n}_k^T - \alpha I)
$$  (8.3)

where $q_k$ is the output from quadrature filter $k$, $\hat{n}_k$ is the orientation of quadrature filter $k$, $I$ is the identity tensor and $\alpha$ is a constant.

The eigenvector $e_1$ of the largest eigenvalue $\lambda_1$ of the tensor $T$ in eq. (8.3) corresponds to the dominant orientation of the neighbourhood. A neighbourhood with a plane having zero acceleration will have one eigenvalue $\lambda_1$ being significantly larger than the other eigenvalues. Other types of neighbourhoods will have different distributions of the eigenvalues.

The filters should be distributed symmetrically over half of the Fourier space. This implies that the filters should be distributed in accordance with the vertices of a regular polytope. The choice is further limited by the restriction that $K$, the number of filters, should be greater than 8 (See Knutsson [60] for a discussion of this requirement). This leaves the 24-cell (Coxeter [30]) as the only alternative. (Computational complexity makes the 120-cell and the 600-cell unrealistic alternatives.)

The frequency domain coordinates for the 12 filter directions are given in appendix 8.B. Appendix 8.C verifies that eq. (8.3) results in the desired representation and determines the value of $\alpha$ (eq. (8.3)) to $\frac{1}{3}$.

### 8.3 Visualization

The estimates produced by the orientation algorithm outlined in this chapter can be seen as ten scalar values jointly describing the situation for each neighbourhood. This type of data can not be visualized using standard techniques. The following method is proposed for visualization of the results, where the data representation is projected into a complex number.

1. Choose two indices $k$ and $l$. 

120
2. Compute the complex number $x^{-1}[t_{kk} - t_{ll}] + i2t_{kl}$ for the 4D-orientation data set. This corresponds in some sense to the 'double angle' representation (chapter 2.1).

3. Map the complex numbers to the colour circle, letting the magnitude determine the intensity and the argument determine the colour.

4. Visualize the colour coded data set with a 'standard' visualization technique.

Six different 'representation-projections' of the above kind can be done. Positions in the data set where the projection, chosen by the indices, fits will have high intensity, and the colour will indicate the '2D-orientation' for the index-pair. Positions where data and projection do not match will have low intensity.

This is the 4D counterpart of the 3D visualization method described in chapter 5.4. The general method with arbitrary orientation of the colour circle (chapter 5.4.1) can of course also be used.

The so called 'standard' visualization techniques deserve a few comments. Two different techniques have been used for the evaluation of the algorithm implementation.

The more general of the methods is volume time sequences, where volume rendering is used to visualize three dimensions and the fourth dimension is represented by time.

The second method is to use a small 2D coordinate system within a large 2D coordinate system, i.e. ordering of the $\xi_1-\xi_2$ slices in a $\xi_3-\xi_4$ mesh (see fig. 8.2). This less sophisticated method does work very well for simple 4D-shapes and gives some insight in the properties of 4D. Both of the techniques are used to visualize the 4D images in this document.
Figure 8.2: A simple 4D visualization technique with a coordinate system within the coordinate system. Note that the spatial coordinate $x$ in the figure corresponds to $\xi_1$ in the text. Likewise, $y = \xi_2$, $z = \xi_3$, $w = \xi_4$.

8.4 Results

Two examples of processing on synthetic 4D data are given, one ‘sphere’ and one ‘cylinder’.

The quadrature filters used were designed in the Fourier domain using eq. (8.9) with

$$H_p(u) = \exp\left(-\frac{1}{\ln 2} \ln^2(2u/\pi)\right), \quad u \in [0..\pi]. \quad (8.4)$$

This is a so called lognorm filter function with centre frequency $\pi/2$ and a bandwidth of 2 octaves. This radial function is plotted in fig. 8.3.

The spatial representation of the quadrature filters were obtained by 4D DFT. Hardware limitations constrained the filters to a maximum size of the neighbourhood to $7 \times 7 \times 7 \times 6$.

8.4.1 The 4D Sphere

The points located on the 4D sphere fulfil the equation

$$\sum_{i=1}^{4} \xi_i^2 = R^2 \quad (8.5)$$

The sphere is visualized in fig. 8.4 using the technique of fig. 8.2. The sphere can be thought of as a time sequence, starting with an empty volume, where a point appears and expands to a huge ball, after which it shrinks back into nothingness.

The sphere used in the experiments has been generated with a smooth surface, i.e. equality in eq. (8.5) is not necessary.

Visualization of the result of the orientation estimation is displayed in fig. A.16–A.21 using the technique described in section 8.3.
Figure 8.3: A lognorm filter with centre frequency $\pi/2$ and a bandwidth of 2 octaves. The plot unit is degrees.

Figure 8.4: The 4D sphere visualized with a coordinate system within the coordinate system.
Fig. A.16 visualizes the orientation in terms of the projection plane corresponding to the small coordinate system, while fig. A.21 visualizes the orientation in terms of the projection plane corresponding to the large coordinate system. Fig. A.17–A.20 are all displaying the orientation in terms of a representation projection plane corresponding to one 'small' coordinate and one 'large' coordinate.

Note how positions in the data set where the projection, chosen by the indices, fits have high intensity. Positions where data and projection do not match have low intensity.

8.4.2 The 4D Cylinder

The name '4D cylinder' will here denote a surface which fulfils the equation

\[ \nu_1^2 + \nu_2^2 = R^2 \]  

(8.6)

where \((\nu_1, \nu_2, \nu_3, \nu_4)\) are cartesian coordinates for a coordinate system \(\nu\) not necessarily coinciding with the \(\xi\) system. Note that there exists a matrix \(Q\) which fulfils the equation

\[ Q\xi = \nu. \]  

(8.7)

This type of surface has one single principal direction and is a suitable test for verification of the algorithm presented in the next chapter. The cylinder used here has the transformation matrix

\[
Q = \begin{pmatrix}
0.8920 & 0.4283 & 0.0162 & -0.1439 \\
0 & 0.1294 & 0.8660 & 0.4830 \\
-0.0290 & -0.2382 & 0.4997 & -0.8323 \\
0.4511 & -0.8620 & 0 & 0.2310
\end{pmatrix}
\]  

(8.8)

which corresponds to four consecutive rotations; \(\pi/6\) in the \((\xi_1, \xi_3)\) plane, \(\pi/4\) in the \((\xi_1, \xi_4)\) plane, \(\pi/3\) in the \((\xi_2, \xi_3)\) plane and \(5\pi/12\) in the \((\xi_2, \xi_4)\) plane.

The cylinder is displayed in fig. 8.5 with the volume time sequence method and the orientation results are collected in fig. A.22.

8.5 Conclusion

Hardware limitations have made the tests of the algorithm quite painful, but the few tests made indicate that the algorithm produces robust and accurate estimates and degrades 'gracefully' in the presence of noise.

Appendices

8.A Filter Shape

The quadrature filters used have the frequency response

\[
\begin{cases}
H_k(u) = H_\rho(u)(\hat{u} \cdot \hat{n}_k)^2 & \text{if } u \cdot \hat{n}_k > 0 \\
H_k(u) = 0 & \text{otherwise}
\end{cases}
\]  

(8.9)
where $\hat{n}_k$ is a unit vector defining the filter direction, $u$ is the frequency with $u$ being the length of the frequency vector and $\hat{u}$ is a unit vector directed as $u$. The output $q_k$ of the corresponding quadrature filter will be a complex number. The magnitude $q_k$ of $q_k$ will be phase invariant (implying local shift invariance) and the argument $\text{arg}(q_k)$ represents the local phase.

In other words, the filter shape is polar separable, the radial part of the function $(H_p(u))$ is arbitrary but positive (usually some type of bandpass function) and the angular part varies as $\cos^2(\varphi)$, where $\varphi$ is the difference in angle between $u$ and the filter direction $\hat{n}_k$.

### 8.B Filter Directions

The 12 filter directions are given in cartesian coordinates by:

\[
\begin{align*}
\hat{n}_1 &= c \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix}^T \\
\hat{n}_2 &= c \begin{pmatrix} 1 & -1 & 0 & 0 \end{pmatrix}^T \\
\hat{n}_3 &= c \begin{pmatrix} 1 & 0 & -1 & 0 \end{pmatrix}^T \\
\hat{n}_4 &= c \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^T \\
\hat{n}_5 &= c \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix}^T \\
\hat{n}_6 &= c \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix}^T \\
\hat{n}_7 &= c \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}^T \\
\hat{n}_8 &= c \begin{pmatrix} 0 & 1 & 0 & -1 \end{pmatrix}^T \\
\hat{n}_9 &= c \begin{pmatrix} 0 & 1 & -1 & 0 \end{pmatrix}^T \\
\hat{n}_{10} &= c \begin{pmatrix} 0 & 1 & 0 & -1 \end{pmatrix}^T \\
\hat{n}_{11} &= c \begin{pmatrix} 0 & 0 & 1 & 1 \end{pmatrix}^T \\
\hat{n}_{12} &= c \begin{pmatrix} 0 & 0 & 1 & -1 \end{pmatrix}^T \\
\end{align*}
\]
where \( c = \frac{1}{\sqrt{2}} \).

### 8.C Filter Combination

This section verifies that eq. (8.3) results in the desired representation. The value of \( \alpha \) (eq. (8.3)) is determined to \( \frac{1}{3} \) by the derivation. The analysis will deal only with real valued neighbourhoods of *one-dimensional variation*, i.e. neighbourhoods that can be expressed as

\[
f(\xi \cdot \hat{n}_s)
\]

where \( \xi \) is the spatial coordinate and \( \hat{n}_s^T = (x_1, x_2, x_3, x_4) \) is a unit vector oriented along the axis of maximal signal variation.

For this type of signal the Fourier transform is non-zero only on the line defined by

\[
u \propto \hat{n}_s
\]

Thus the situation can be treated as 1-dimensional and, using eq. (8.9) and 1-dimensional filter theory, it is not hard to show that the magnitude of the quadrature filter output (as a function of the signal orientation) is given by

\[
q_k = d(\hat{n}_s \cdot \hat{n}_k)^2
\]

where \( d \) can be considered a constant, as it is independent of the filter orientation and depends only on the magnitude and radial distribution of the signal spectrum.

Insertion of eq. (8.10) and eq. (8.13) into eq. (8.3) is fairly straightforward. The elements of the \( \hat{n}_k \hat{n}_k^T \)'s are given by

\[
\hat{n}_1 \hat{n}_1^T = c^2 \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_2 \hat{n}_2^T = c^2 \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_3 \hat{n}_3^T = c^2 \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_4 \hat{n}_4^T = c^2 \begin{pmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_5 \hat{n}_5^T = c^2 \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}
\]

126
The magnitude of the outputs from the 12 quadrature filters are given by

\[
\hat{n}_6\hat{n}_6^T = c^2 \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}
\]

\[
\hat{n}_7\hat{n}_7^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_8\hat{n}_8^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}
\]

\[
\hat{n}_9\hat{n}_9^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}
\]

\[
\hat{n}_{10}\hat{n}_{10}^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix}
\]

\[
\hat{n}_{11}\hat{n}_{11}^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}
\]

\[
\hat{n}_{12}\hat{n}_{12}^T = c^2 \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}
\]
Next, calculating the sum

\begin{align}
q_1 &= d c^2 x^2 (x_2^2 + 2 x_1 x_2 + x_2^2) \\
q_2 &= d c^2 x^2 (x_2^2 - 2 x_1 x_2 + x_2^2) \\
q_3 &= d c^2 x^2 (x_1^2 + 2 x_1 x_2 + x_2^2) \\
q_4 &= d c^2 x^2 (x_2^2 - 2 x_1 x_3 + x_3^2) \\
q_5 &= d c^2 x^2 (x_1^2 + 2 x_1 x_4 + x_4^2) \\
q_6 &= d c^2 x^2 (x_2^2 - 2 x_1 x_4 + x_4^2) \\
q_7 &= d c^2 x^2 (x_2^2 + 2 x_2 x_3 + x_3^2) \\
q_8 &= d c^2 x^2 (x_2^2 - 2 x_2 x_3 + x_3^2) \\
q_9 &= d c^2 x^2 (x_2^2 + 2 x_2 x_4 + x_4^2) \\
q_{10} &= d c^2 x^2 (x_2^2 - 2 x_3 x_4 + x_4^2) \\
q_{11} &= d c^2 x^2 (x_3^2 + 2 x_3 x_4 + x_4^2) \\
q_{12} &= d c^2 x^2 (x_3^2 - 2 x_3 x_4 + x_4^2) \\
\end{align}

(8.15)

where \(d^' = 2d\).

It is evident that if the quantity \(\frac{1}{2} d^'\) is subtracted from the diagonal elements of \(T''\) the result will be of the desired form.

\begin{align}
T'' &= \sum_k q_i \hat{n}_k \hat{n}^T_k \\
T' &= T'' - \frac{1}{2} d^' = d' \hat{n}_s \hat{n}^T_s \\
\end{align}

(8.16)
Finally calculate the sum of all quadrature filter output magnitudes.

\[ \sum_k q_k = 3d \]  

(8.19)

Combining eqns. (8.16), (8.18) and (8.19) yields the desired result:

\[ T'(d' \hat{n}_k) = \sum_k q_k(\hat{n}_k \hat{n}_k^T - \frac{1}{3}I) \]  

(8.20)
Chapter 9

Curvature Estimation in 4D

It is easy enough to make faulty assumptions and mistakes when one is dealing with the geometry of three dimensions\(^1\). This is somewhat peculiar since it seems that the experience of living in three space would give the necessary insights.

The nature of Four-dimensional space is on the other hand nothing that one is naturally familiar with. The problem of 4D visualization taken together with the computational complexity of the algorithms makes it hard to test and evaluate the performance and correctness of the methods.

One of the components of the curvature model is rotation. This operation is trivial in 2D and well known in 3D with procedures such as rotation matrices, quaternions, etc. Four-dimensional rotation is less intuitive. Such a rotation takes place in a 2D plane and has not one but two rotation axes, which at first may seem a bit disturbing.

9.1 The 4D Principal Direction Algorithm

The principal direction algorithm is readily modified for four-dimensional data.

9.1.1 Input Data

Note that six 'double angle' complex values can be obtained from the orientation tensor.

\[
\begin{align*}
(x_1^2 - x_2^2) + i2x_1x_2 & \\
(x_1^2 - x_3^2) + i2x_1x_3 & \\
(x_1^2 - x_4^2) + i2x_1x_4 & \\
(x_2^2 - x_3^2) + i2x_2x_3 & \\
(x_2^2 - x_4^2) + i2x_2x_4 & \\
(x_3^2 - x_4^2) + i2x_3x_4 &
\end{align*}
\]

Eqns. (9.1), (9.2) and (9.4) are the same as for the 3D tensor (eqns. (5.10)-(5.12)).

\(^1\)Readers with a different opinion are recommended to read [67, 68]
9.1.2 Shift Estimation

The shift theorem of the Fourier Transform is valid in 4D.

\[ F_1(u) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} r(\xi) \exp(2\pi i t \cdot \xi) \exp(-2\pi i u \cdot \xi) d\xi = R(u - t) \]  
\[ \varphi(\xi) = 2\pi(t \cdot \xi) \]

where the vector \( t \) is directed along the dominant direction.

And these two equations, eq. (9.7) and (9.8), imply, in accordance with the 2D and 3D counterparts, that estimates of the centre of gravity of the local Fourier spectra estimate the curvature for this type of neighbourhood.

The same shift estimation algorithm can be used, i.e. filtering the input data with symmetrically distributed filters combined with a vector summation of the filter responses. A preferable choice of 4D body for the Fourier domain distribution of the filters is the 16-cell (see fig. 9.1). The coordinates for the Fourier domain directions of the filters are given in appendix 9.A. It is demonstrated in appendix 9.C that this filtering realization only requires five scalar 4D filters. Keeping this number low is essential since the input data consists of ten different scalar fields. The 16-cell will thus require 50 4D convolutions, a number which is even worse for other 4D bodies.

The quasi-quadrature filter is the same as in 2D and 3D, i.e. polar separable with
the radial function $H_\theta$ being a bandpass filter and the angular function equals

$$
\begin{align*}
H_\theta(u) &= \cos^2 \frac{\theta}{2} \\
\theta &= \arccos(\hat{u} \cdot \hat{n}_k)
\end{align*}
$$

(9.9)

One difference is, however, that it is necessary to use a slightly modified vector summation formula to obtain a shift estimate suitable for the principal direction representation. The new formula is

$$
\begin{align*}
b_i'' &= \sum_{k=1}^{K} q_k^2 n_k \\
b_i' &= \sqrt{b_i'' b_i'''}
\end{align*}
$$

(9.10)

The reason for this slightly modified vector summation is given in the proof given in appendix 9.E.

9.1.3 The Representation

The representation consists of six 4D-vectors, obtained by using eq. (9.10) on the six complex-valued data sets obtained by eqns. (9.1)-(9.6). It has been discussed that the shift of the centre of gravity of the Fourier spectra is correctly estimated by eq. (9.10). This is but one of the items on the list of criteria which is necessary to fulfill to have a suitable representation of the principal directions and an algorithm which produces this representation. The rest of the list is as follows:

- The data-projection from tensor to complex number results in a Fourier spectrum, where the direction of the shift coincides with the principal direction.
- The representation is rotational invariant.
- The representation is continuous.
- The representation can handle three principal directions.

The first item on the list above concerns the algorithm, while the other items concern the representation. The appendices at the end of this chapter demonstrate and discuss that these requirements are fulfilled. Note from the list above that a 3D hypersurface embedded in a flat 4D space has three principal curvatures $\kappa_1$, $\kappa_2$ and $\kappa_3$ with principal directions of curvature $t_1$, $t_2$ and $t_3$ [59].

The representation of the principal directions of curvature is:

$$
\begin{align*}
b_1' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^4 - n_i^2 m_i^2) t_i \\
b_2' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^2 - n_i^2 m_i^4) t_i \\
b_3' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^3 - n_i^2 m_i^1) t_i \\
b_4' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^4 - n_i^2 m_i^2) t_i \\
b_5' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^1 - n_i^2 m_i^3) t_i \\
b_6' &= \sum_{i=1}^{3} X_i (n_i^2 m_i^2 - n_i^2 m_i^4) t_i
\end{align*}
$$

(9.11)

where $n^i$ and $m^i$ are the two rotation axes of $t_i$. The magnitude of the scalar $X_i$ depends on the energy (certainty) of the orientation estimates in the neighbourhood and the fit between the principal curvature $\kappa_i$ and the frequency characteristics of the filters used.
9.1.4 From the Representation to the Principal Directions

The principal directions are obtained by computing the eigenvalues and eigenvectors of

\[ \sum_{i=1}^{6} b_i^j b_i^k \]  \hspace{1cm} (9.12)

9.2 Results

The examples of processing on synthetic 4D data are the same as in chapter 8, the 'sphere' and the 'cylinder'.

The quasi-quadrature filters used to filter the orientation tensor data were designed in the Fourier domain with a lognorm function as the radial part of the polar separable filter function. The spatial representation of the quasi-quadrature filters were obtained by 4D DFT. Hardware limitations constrained the filters to a maximum size of the neighbourhood to 7 x 7 x 7 x 6.

The output from the principal direction algorithm consists of six 4D-vectors. Each such 4D-vector is separated into two 2D-vectors for visualization. The scheme for display of 4D orientation described in chapter 8.3 is used with some modifications. Intensity and colour are used to represent magnitude and argument of the complex numbers (i.e., 2D vectors) in combination with the technique where a coordinate system is located within the coordinate system (fig. 8.2). The differences are that two such 'meta-images' are needed, one for each 2D vector, and that the colour here represents principal directions of curvature and not orientation.

The output of the principal direction estimation is displayed with the \( b_1 + ib_2 \) parts to the left and the \( b_3 + ib_4 \) parts to the right.

9.2.1 The 4D Sphere

The orientation data was resampled before the use of the principal direction algorithm. Reducing each dimension by a factor of two reduces the 4D data set to 1/16 of the original spatial dimensions. The benefit of this quite substantial reduction is obvious when one considers the fact that the orientation information consists of ten scalar values per data element. Two different \( b_i \)'s are shown in each figure (fig. A.23–A.25).

9.2.2 The 4D Cylinder

The 4D cylinder has one single principal direction, and the six 4D vectors produced by the principal direction algorithm should for this type of data be directed along one single line. The vectors are pairwise either equally or opposite directed. This is verified by examination of fig. A.26.

9.3 Conclusion

This chapter has presented the generalization of the principal direction algorithm to 4D. The use of the output from this algorithm, such as the estimation of the sign of
the gaussian curvature, has not been explored, partly due to hardware limitations. However, the discussion in this chapter taken together with the results of processing on synthetic data indicates that the principal direction algorithm has the potential of being quite useful in the processing of 4D images when such data becomes available.

Appendices

9.A Fourier Domain Directions of the Filters

The filters are distributed as the vertices of a 16-cell (fig. 9.1).

\[ n_1 = (1, 0, 0, 0)^T \]  
\[ n_2 = (0, 1, 0, 0)^T \]  
\[ n_3 = (0, 0, 1, 0)^T \]  
\[ n_4 = (0, 0, 0, 1)^T \]  
\[ n_5 = (-1, 0, 0, 0)^T \]  
\[ n_6 = (0, -1, 0, 0)^T \]  
\[ n_7 = (0, 0, -1, 0)^T \]  
\[ n_8 = (0, 0, 0, -1)^T \]

9.B Rotation Invariant Shift Estimation

The eight filters have, for an impulse located at \( \rho(\cos \alpha, \cos \beta, \cos \gamma, \cos \delta) \), the response

\[ q_1 = H_\rho(\rho)/2(1 + \cos \alpha) \]  
\[ q_2 = H_\rho(\rho)/2(1 + \cos \beta) \]  
\[ q_3 = H_\rho(\rho)/2(1 + \cos \gamma) \]  
\[ q_4 = H_\rho(\rho)/2(1 + \cos \delta) \]  
\[ q_5 = H_\rho(\rho)/2(1 - \cos \alpha) \]  
\[ q_6 = H_\rho(\rho)/2(1 - \cos \beta) \]  
\[ q_7 = H_\rho(\rho)/2(1 - \cos \gamma) \]  
\[ q_8 = H_\rho(\rho)/2(1 - \cos \delta) \]

Computation of \( b'' \) (eq. (9.10)) yields

\[ b'' = H^2_\rho(\rho)(\cos \alpha, \cos \beta, \cos \gamma, \cos \delta) \]  
and consequently

\[ b' = H_\rho(\rho)(\cos \alpha, \cos \beta, \cos \gamma, \cos \delta) \]
9.C Obtaining Eight Complex Filter Responses by Five Scalar Filters

Examination of \( H_1(u) \) gives

\[
H_1(u) = H_\rho(u) \cdot H_\theta = \frac{H_\rho(u)}{2} + \frac{H_\rho \cos \alpha}{2} = \tag{9.31}
\]

Note that \( H_\rho(u) \) is an even function and that \( H_\rho(u) \cos \alpha \) is an odd function. This implies that the spatial representation of \( H_1 \) is Hermitian (Bracewell [27]), i.e.

\[
h_1(\xi) = e_1(\xi) + io_1(\xi) \tag{9.32}
\]

where \( e_1 \) is the even part of \( h_1 \) and \( o_1 \) is the odd part.

Observe that

\[
e_1(\xi) = \frac{h_\rho(\xi)}{2} = e_2(\xi) = \ldots = e_8(\xi). \tag{9.33}
\]

That is, the real parts of the complex valued filters \( h_i \) are all identical (and of Laplace-type). It is thus sufficient to convolve with one filter to obtain the response from the real parts of the eight filters.

The second observation that can be made is

\[
\begin{align*}
e_1(\xi) &= -e_5(\xi) \\
e_2(\xi) &= -e_6(\xi) \\
e_3(\xi) &= -e_7(\xi) \\
e_4(\xi) &= -e_8(\xi)
\end{align*} \tag{9.34}
\]

This implies that the convolution with the imaginary parts of the eight filters can be obtained by four convolutions and sign changes.

It is consequently enough to perform five \((1 + 4)\) scalar convolutions to obtain eight complex convolutions.

9.D The Direction of Fourier Spectra Shift Coincides with the Principal Direction

Assume that the neighbourhood contains one single principal curvature. The normal vectors of the 4D-surface can for this type of neighbourhood be approximated by

\[
\begin{align*}
v_1 &= a_1 \cos \theta + c_1 \sin \theta \\
v_2 &= a_2 \cos \theta + c_2 \sin \theta \\
v_3 &= a_3 \cos \theta + c_3 \sin \theta \\
v_4 &= a_4 \cos \theta + c_4 \sin \theta
\end{align*} \tag{9.35}
\]

where \( v = (v_1, v_2, v_3, v_4) \) is the normal vector located at a point with (local) coordinates \( \xi \). \( a \) is the normal vector of the neighbourhood origin and \( c \) is a vector orthogonal to \( a \). The vectors \( a \) and \( c \) span a 2D sub-space of the 4D vector space.

The parameter \( \theta \) is computed as \( \kappa_1(t_1, \xi) \). \( t_1 \) denotes the principal direction, \( \kappa_1 \) denotes the amount of curvature and \( \xi \) the (local) coordinate.

This corresponds to a 4D neighbourhood with vectors rotating around two axes \( m \) and \( n \), and the formulation above (eq. (9.35)) is the counterpart of the linear approximation of curvature used in 2D in chapter 2 (eq. (2.10)) and in 3D in chapter 6 (eq. (6.19)).
Changes in length (magnitude) of the vectors do not influence the model, since magnitude changes and angular changes are separated in the model formulation (eq. (9.7) and (9.8)). It is for that reason no loss of generality to assume that the vectors \( a, c, m \) and \( n \) are of unit length.

Inspection of one of the data projections, e.g. \( (v_1, v_2) \), will reveal both similarities and differences to the 3D algorithm, but first a short repetition of the neighbourhood model, which requires the same modification as in 3D (chapter 6). The model is still polar separable with

\[
\Phi(\xi) = r(\xi) \Phi(\xi)
\]

where \( r(\xi) \) describes the changes in magnitude \( \sqrt{v_1^2 + v_2^2} \) and \( \Phi(\xi) \) describes the angular changes. An examination of \( (v_1, v_2) \) with the objective to obtain an expression for \( \Phi(\xi) \) where it is possible to utilize the shift theorem gives

\[
\begin{pmatrix}
v_1 \\
v_2
\end{pmatrix} = \begin{pmatrix}
a_1 \cos \theta + c_1 \sin \theta \\
a_2 \cos \theta + c_2 \sin \theta
\end{pmatrix} = \begin{pmatrix}
\frac{a_1 + c_2}{2} \cos \theta - \frac{a_2 - c_1}{2} \sin \theta \\
\frac{a_1 + c_2}{2} \sin \theta + \frac{a_2 - c_1}{2} \cos \theta
\end{pmatrix} + \begin{pmatrix}
\frac{a_1 - c_2}{2} \cos \theta + \frac{a_2 + c_1}{2} \sin \theta \\
\frac{a_1 - c_2}{2} \sin \theta - \frac{a_2 + c_1}{2} \cos \theta
\end{pmatrix} = (9.37)
\]

This implies that \( \Phi(\xi) \) can be written as

\[
\Phi(\xi) = \sqrt{\left(\frac{a_1 + c_2}{2}\right)^2 + \left(\frac{a_2 - c_1}{2}\right)^2 \exp(i\theta(\xi))} + \sqrt{\left(\frac{a_1 - c_2}{2}\right)^2 + \left(\frac{a_2 + c_1}{2}\right)^2 \exp(-i\theta(\xi))}
\]

(9.38)

The projection of the 4D orientation into a complex number thus results into not one but two shifts in the Fourier domain of the local neighbourhood. These two shifts are opposite directed and eq. (9.10) produces a vector pointing in the direction of the strongest one.

### 9.E Rotation Invariance

This criterion can be expressed as

\[
\sum_{i=1}^{6} (b'_i)^2 = X^2
\]

(9.39)

where \( b'_i \) denotes the magnitude of one of the six 4D-vectors describing the local Fourier spectra shifts of the complex-valued data set (eqns. (9.1)-(9.6)).

\( X \) only depends on the amount of curvature, the certainty of the orientation estimates and the frequency characteristics of the quasi-quadrature filter used. That is, \( X \) is invariant to the principal direction \( t_1 \).

Returning to eq. (9.38) and its Fourier shifts and combining it with eq. (9.10) gives that the ‘positive’ shift will result in a vector pointing in the \( t_1 \) direction with the magnitude

\[
Y \cdot \left(\frac{(a_1 + c_2)^2}{2} + \frac{(a_2 - c_1)^2}{2}\right)
\]

(9.40)
The 'negative' shift will likewise produce a vector pointing in the \(-t_1\) direction with magnitude
\[
Y \cdot \left( \frac{a_1 - c_2}{2} \right)^2 + \left( \frac{a_2 + c_1}{2} \right)^2
\]  
(9.41)
where \(Y\) depends on the frequency characteristics of the quasi-quadrature filter together with the shape of \(R(u)\), the local Fourier transform of the magnitude function \(r(\xi)\) (eq. (9.7)).

Summation of the two opposite directed vectors of eq. (9.38) gives \(b'_1\) of eq. (9.10) with a magnitude equal to
\[
b'_1 = Y |a_1 c_2 - a_2 c_1|
\]  
(9.42)
An investigation of eq. (9.42) using the following relations derived from the determinant formula for 4D vectors
\[
c_1 = a_2 n_3 m_4 + a_3 n_4 m_2 + a_4 n_2 m_3 - a_2 n_4 m_3 - a_3 n_2 m_4 - a_4 n_3 m_2
\]  
(9.43)
\[
c_2 = a_1 n_4 m_3 + a_3 n_1 m_4 + a_4 n_3 m_1 - a_1 n_3 m_4 - a_3 n_4 m_1 - a_4 n_1 m_3
\]  
(9.44)
gives
\[
a_1 c_2 = a_1^2 n_4 m_3 - a_1^2 n_3 m_4 + a_1 a_3 n_1 m_4 - a_1 a_3 n_4 m_1 + a_1 a_4 n_3 m_1 - a_1 a_4 n_1 m_3
\]  
(9.45)
and
\[
a_2 c_1 = a_2^2 n_3 m_4 - a_2^2 n_4 m_3 + a_2 a_3 n_2 m_4 - a_2 a_3 n_4 m_2 + a_2 a_4 n_2 m_3 - a_2 a_4 n_3 m_2
\]  
(9.46)
Insertion of eq. (9.45) and (9.46) into (9.42) gives
\[
b'_1 = Y \left| (a_1^2 + a_2^2)(n_4 m_3 - n_3 m_4) + a_1 a_3(n_1 m_4 - n_4 m_1) + a_1 a_4(n_3 m_1 - n_1 m_3) + a_2 a_3(n_2 m_4 - n_4 m_2) + a_2 a_4(n_3 m_2 - n_2 m_3) \right|
\]  
(9.47)
Rearranging the terms
\[
b'_1 = Y \left| (a_1^2 + a_2^2)(n_4 m_3 - n_3 m_4) - (a_1 n_1 + a_2 n_2)a_3 m_4 - (a_1 a_1 + a_2 a_2)a_4 m_3 + (a_1 m_1 + a_2 m_2)a_4 m_3 - (a_1 m_1 + a_2 m_2)a_3 n_4 \right|
\]  
(9.48)
Insertion of
\[
a_1 n_1 + a_2 n_2 + a_3 n_3 + a_4 n_4 = 0
\]  
(9.49)
and
\[
a_1 m_1 + a_2 m_2 + a_3 m_3 + a_4 m_4 = 0
\]  
(9.50)
into eq. (9.48) gives
\[
b'_1 = Y \left| (a_1^2 + a_2^2)(n_4 m_3 - n_3 m_4) - (a_3 n_3 + a_4 n_4)a_3 m_4 + (a_3 a_3 + a_4 n_4) a_4 m_3 - (a_3 m_3 + a_4 m_4) a_4 m_4 \right|
\]  
(9.51)
Simplifying
\[
b'_1 = Y \left| (a_1^2 + a_2^2 + a_3^2 + a_4^2)(n_4 m_3 - n_3 m_4) \right|
\]  
(9.52)
and finally using that \(a\) is a unit vector
\[
b'_1 = Y |n_4 m_3 - n_3 m_4|
\]  
(9.53)
Likewise

\[ b'_2 = Y|n_2m_4 - n_4m_2| \]  \hspace{1cm} (9.54)

\[ b'_3 = Y|n_2m_3 - n_3m_2| \]  \hspace{1cm} (9.55)

\[ b'_4 = Y|n_1m_4 - n_4m_1| \]  \hspace{1cm} (9.56)

\[ b'_5 = Y|n_1m_3 - n_3m_1| \]  \hspace{1cm} (9.57)

\[ b'_6 = Y|n_1m_2 - n_2m_1| \]  \hspace{1cm} (9.58)

Using eq. (9.53)–(9.58) to compute the square sum

\[
\sum_{i=1}^{6} (b'_i)^2 = Y^2(n_3m_4 - n_4m_3)^2 + (n_2m_4 - n_4m_2)^2 + (n_1m_4 - n_4m_1)^2 + \\
(n_2m_3 - n_3m_2)^2 + (n_1m_3 - n_3m_1)^2 + (n_1m_2 - n_2m_1)^2 \\
= Y^2[n_3^2m_4^2 + n_4^2m_3^2 - 2n_3m_3n_4m_4n_2^2m_2^2 + n_2^2m_2^2 - 2n_2m_2n_4m_4 \\
+ n_1^2m_1^2 + n_4^2m_4^2 - 2n_1m_1n_4m_4n_2^2m_2^2 + n_2^2m_2^2 - 2n_2m_2n_3m_3 \\
- n_1^2m_1^2 + n_3^2m_3^2 - 2n_1m_1n_3m_3n_2^2m_2^2 + n_2^2m_2^2 - 2n_2m_2n_1m_1] \]  \hspace{1cm} (9.59)

Insertion of

\[
(n_1m_1 + n_2m_2 + n_3m_3 + n_4m_4)^2 = n_1^2m_1^2 + n_2^2m_2^2 + n_3^2m_3^2 + n_4^2m_4^2 + \\
2n_1m_1n_2m_2 + 2n_1m_1n_3m_3 + 2n_1m_1n_4m_4 + \\
2n_2m_2n_3m_3 + 2n_2m_2n_4m_4 + 2n_3m_3n_4m_4 = 0 \]  \hspace{1cm} (9.60)

into eq. (9.59) gives

\[
\sum_{i=1}^{6} (b'_i)^2 = Y^2(n_1^2 + n_2^2 + n_3^2 + n_4^2)(m_1^2 + m_2^2 + m_3^2 + m_4^2) = Y^2 \cdot 1 \cdot 1 = Y^2 \]  \hspace{1cm} (9.61)

This implies rotation invariance$^2$.

### 9.F Continuous Representation

It was earlier demonstrated that the principal direction of curvature $t_1$ coincides with the direction of either $b'_i$ or $-b'_i$ for all $i$. Continuous changes of the direction $t_1$ will consequently change the direction of $b'_i$ (and possibly also smooth changes of the magnitude $b_i$). The change of $b'_i$ will be continuous if no ‘sudden’ sign changes occur.

Examination of eq. (9.10), (9.40) and (9.41) gives that the sign changes occur when

\[ a_je_k = a_kc_j. \]  \hspace{1cm} (9.62)

This does not create any discontinuities since the magnitude $b_i$ at this instance equals zero (eq. (9.42)).

### 9.G Three Principal Directions

The principal directions of curvature, $t_1$, $t_2$ and $t_3$ are orthogonal to each other. This ensures that the different $t_i$ do not interfere with each other in the vector summation in eq. (9.10).

$^2$See Waterson [96, p. 244] for a comment regarding this derivation.
Chapter 10

Summary and Discussion

A methodology for curvature estimation and description in computer vision and image processing applications has been presented. The algorithms described are in many respects different to other methods. The most significant is that the method gives a unified approach to the estimation and description of orientation and curvature. The concept of curvature and the description thereof in the interpretation given here become very similar to the methods for orientation description. In addition, the methodology avoids thresholding and premature decision making.

Emphasis has been placed on frequency domain features, and a model of how curvature is reflected in the Fourier domain has been proposed. Different types of multi-dimensional data have been considered: images (2D); volumes (3D); time sequences of images (3D); and time sequences of volumes (4D).

Results on both synthetic and real-world data have been evaluated, and some comparisons with other algorithms have been carried out. The results indicate that the methodology presented here has a number of important advantages over other methods.

10.1 Future Work

It is easy to make a list of items for further development of the algorithms:

- The 2D algorithms should be incorporated into more applications than the ones described in chapter 4.
- The methods described in chapter 4 should be developed into full-fledged applications.
- The 3D ‘amount of curvature’ estimation algorithm using different scales should be fully implemented and evaluated.
- The relationship between the signal-processing approach to curvature used in this thesis and ‘true’ differential geometry should be more thoroughly examined.
- The acceleration estimation should use a more fully developed mathematical model.
- The 4D algorithms should be evaluated on real-world data.
10.2 Curvature Mechanisms in the Human Visual System

The Human Visual System (HVS) has earlier been listed as one of the sources of inspiration for the methodology presented in this thesis. This chapter ends with a discussion considering the relationship between the algorithms presented in earlier chapter and evidence concerning the mechanisms in the HVS.

Although it is presently impossible to infer precisely how the HVS extracts curvature information, there exist numerous indications from the fields of perception and psychophysics as well as neurobiology of how the procedure may function (e.g. Hubel and Wiesel [52], Blakemore and Over [25], Timney and Macdonald [93], Wilson [99], Zucker [105], Wilson and Richards [100]).

One of the major breakthroughs in neurobiology was the discovery by Hubel and Wiesel of orientation-selective cells (see e.g. [53] for an introduction to this area). The striate cortex with its manifold features has been of great interest ever since. One of the most striking features is the organization of orientation-selective cells into 'orientation-columns'. The response of quadrature filters resembles the response of the orientation selective cells in the striate cortex (Daugman [32]). Neural network algorithms which develop orientation selective cells in quadrature pairs have been described (e.g. Yuille et al [103]).

One of the more popular theories for the HVS curvature mechanism is a scheme where the curvature information is integrated from the output from the orientation detectors in the visual cortex. The parallels between this psychophysics theory and the computer vision algorithm for curvature estimation proposed in this thesis are obvious. Further on, recall that the algorithm uses line (or laplace) and edge filters on the orientation data (eq. (2.16)). It is well known that these types of filters are used in the HVS to extract the orientation information [53].

It is noteworthy that both the orientation and the curvature algorithm have mechanisms which have been proposed in the psychophysics research field, namely lateral inhibition between the orientation/curvature detectors. Incompatible orientations and curvatures are opposite directed in the vector summations in eq. (2.5) and (2.14). This can be compared with the lateral inhibition between orientation detectors discussed in Blakemore et al [24] and Blakemore and Tobin [26], and the excitatory and inhibitory regions for curvature discussed in Wilson and Richards [100, fig. 11b].

The above mentioned argument makes the curvature direction algorithm one of the candidates for the possible mechanisms in the HVS integrating the orientation information into curvature information. However, to the best of my knowledge there are no neurobiological results speaking in favour for or against this theory. A physiological evaluation would require simultaneous recording of the activity in a group of neurons: the orientation selective cells and the hypothetical curvature direction cells integrating the output from the orientation cells. It is also difficult to design a psychophysical perception experiment which would test the validity of this hypothesis.
Appendix A

Colour Images

The colour images in this appendix belong to the following chapters:

- Chapter 2: fig. A.1–A.7
- Chapter 4: fig. A.8
- Chapter 5: fig. A.9–A.10
- Chapter 6: fig. A.11–A.13
- Chapter 7: fig. A.14–A.15
- Chapter 8: fig. A.16–A.22
- Chapter 9: fig. A.23–A.26
Figure A.1: Upper image: The test pattern. Middle image: Curvature direction with thresholded magnitude. Lower image: Curvature/linearity ratio.

Figure A.2: The result from the curved/straight algorithm on the circle image. Red=curved, blue=mixed, green=straight.
Figure A.3: Discrimination of the different circles with use of the curved/straight ratio

Figure A.4: Orientation estimate from 'Lenna'. The colour code is given in fig. 2.10 and 2.16
Figure A.5: Curvature direction estimate from 'Lenna'. The representation is ex-amplified in fig. 2.10

Figure A.6: Curvature magnitude estimate from 'Lenna'. Yellow=low, green=medium, blue=high
Figure A.7: Curved/straight estimate from 'Lenna'. Red=curved, blue=mixed, green=straight

Figure A.8: Top: Original orientation image. Middle: Vector average. Bottom: Vector enhancement
Figure A.9: Example of 3D orientation visualization with a matrix of tensor element images. Green and red colour indicate positive and negative values respectively.

Figure A.10: Example of 3D orientation visualization. The orientation tensor is interpreted as three 'double angle' 2D-vectors. Note the interpretation spheres displayed below the corresponding sub-image.
Figure A.11: The local shape estimates

Figure A.12: The output from the straight-forward differential geometry approach. Green colour indicates positive Gaussian curvature and red colour negative Gaussian curvature.
Figure A.13: The output from the eggvalue algorithm. Green colour indicates ellipsoid surface patches and red colour indicates saddle patches

Figure A.14: Upper left: Circle moving with constant velocity. Upper right: Acceleration direction estimate. Lower left: Circle accelerating to the left. Lower right: Estimate. Red colour = acceleration to the left
Figure A.15: A frame from the bungy-jump sequence. The superimposed acceleration direction estimate has good correspondence to the apparent ground truth.

Figure A.16: The $x_1 x_2$ representation projection of the orientation tensor for the 4D sphere. The colour indicates the '2D–orientation' for the index-pair.
Figure A.17: The $x_1x_3$ representation projection of the orientation tensor for the 4D sphere.

Figure A.18: The $x_1x_4$ representation projection of the orientation tensor for the 4D sphere.
Figure A.19: The $x_2 x_3$ representation projection of the orientation tensor for the 4D sphere.

Figure A.20: The $x_2 x_4$ representation projection of the orientation tensor for the 4D sphere.
Figure A.21: The $x_3 x_4$ representation projection of the orientation tensor for the 4D sphere.

Figure A.22: The six representation projections of the orientation tensor for the 4D cylinder
Figure A.23: The 4D sphere. Upper part: \( b'_1 \), i.e., data obtained by processing on the \( x_1x_2 \)-projection. Lower part: \( b'_0 \) obtained by processing on the \( x_3x_4 \)-projection. The output of the principal direction estimation is displayed with \( b'_{11} + ib'_{12} \) parts to the left and \( b'_{13} + ib'_{14} \) parts to the right.
Figure A.24: The 4D sphere. Upper part: $b'_2$, obtained by processing on the $x_1x_3$-projection. Lower part: $b'_3$ obtained by processing on the $x_2x_4$-projection. Both with $b'_{i1} + ib'_{i2}$ to the left and $b'_{i3} + ib'_{i4}$ to the right.

Figure A.25: The 4D sphere. Upper part: $b'_4$, obtained by processing on the $x_1x_4$-projection. Lower part: $b'_4$ obtained by processing on the $x_2x_3$-projection. Both with $b'_{i1} + ib'_{i2}$ to the left and $b'_{i3} + ib'_{i4}$ to the right.
Figure A.26: The representation of the principal direction of the 4D cylinder. All of the $b'$-vectors are parallel, either equally or opposite directed. This is easily verified in this figure.
Bibliography


159


[92] the Bible. the epistle to the ephesians iii, 17–18.


