Comparison of structure tensors for direction and motion estimation in medical ultrasound

Ole-Christian Schmidt Hagenes

Thesis submitted for the degree of
Master in Informatics: Nanoelectronics and Robotics,
Signal Processing
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019
Comparison of structure tensors for direction and motion estimation in medical ultrasound

Ole-Christian Schmidt Hagenes
Abstract

Medical ultrasound is a non-invasive low-cost image modality used in diagnostic applications. It has become the preferred modality in cardiology. Motion estimation techniques may be applied to the echocardiographic images to estimate 2D blood motion and measure myocardial strain. Optical flow techniques may be used, but the choice of algorithm need to be carefully selected to achieve the required robustness and accuracy.

This thesis compares the performance of structure tensor based methods for calculating the optical flow motion estimation in synthetic and optical images, as well as in realistic synthetic echocardiographic image sequences. The methods for calculating the structure tensor are gradients, Riesz transform and quadrature filters. Estimation of optical flow from the structure tensors is done by Lucas-Kanade and the eigenvector method.

The Riesz transform is an extension of the Hilbert transform to higher dimensions. It is shown that it extracts the same information as the gradient, but without the high frequency amplification.

Experiments are performed on a synthetic no-noise images, the Middlebury dataset containing both synthetic and optical images, and on realistic synthetic echocardiographic images. The Riesz and quadrature tensor are found to outperform the gradient in homogeneous regions in low-noise images. However, on the realistic synthetic echocardiographic images they perform equal with the gradient. Thus no significant evidence for replacing the gradient with the Riesz transform or quadrature filters for motion estimation in echocardiography are found.
Acknowledgements

This thesis is submitted as a part of the master’s degree in informatics: Nanoelectronics and Robotics, Signal Processing. It has been very interesting and educational to work with optical flow on medical ultrasound.

I would especially like to thank my supervisor Sten Roar Snare for his advice, helpful comments and for granting me patience the last and most hectic weeks.

Thanks to my internal supervisor Professor Anne H. Solberg and the rest of the DSB group¹, for good help and an inviting atmosphere. And thanks for many insightful discussions.

Finally, I would like to thank my parents for their encouragement and everlasting love.

¹Digital Signal Processing and Image Analysis group at the department of Informatics (IFI), University of Oslo
Contents

Abstract v

Acknowledgements vii

1. Introduction 1

2. Background 3
   2.1. Optical flow ................................................................. 3
      2.1.1. Introduction .......................................................... 3
      2.1.2. Lucas Kanade ....................................................... 8
      2.1.3. Horn Schunk ......................................................... 10
      2.1.4. Limitations ......................................................... 12
   2.2. Space-time images ..................................................... 13

3. Monogenic Signal 15
   3.1. 1D signal and the analytic signal .................................... 15
      3.1.1. The Hilbert transform ............................................. 15
      3.1.2. The analytic signal ............................................... 16
      3.1.3. Properties of the analytic signal ................................ 16
   3.2. nD signals ................................................................. 17
      3.2.1. Intrinsic dimensionality ......................................... 17
   3.3. Extension of the analytic signal to higher dimensions ................. 18
      3.3.1. Total Hilbert transform .......................................... 19
      3.3.2. Partial Hilbert transform ....................................... 19
      3.3.3. Riesz transform .................................................. 20
   3.4. Monogenic signal ....................................................... 21

4. Structure Tensor 23
   4.1. Steerable filters ......................................................... 23
      4.1.1. Introduction .......................................................... 23
      4.1.2. Generalized directional derivatives .............................. 23
   4.2. Structure tensor .......................................................... 25
      4.2.1. Oriented signal energy ........................................... 25
      4.2.2. Gradient tensor .................................................... 25
      4.2.3. Quadrature tensor ................................................ 26
## Contents

9.2. **Future work** ......................................................... 57  
9.2.1. Region based error ................................................. 57  
9.2.2. Adaptive parameter selection .................................... 57  
9.2.3. Space-time implementation of Riesz transform ............... 58  
9.2.4. Local warp ......................................................... 58  
9.2.5. Complex motion models ........................................ 58  
9.2.6. Use of different pre-smoothing filters for the Riesz transform ........................................ 58  

A. **Parameter tuning** ..................................................... 59  
   A.1. Square boxes ....................................................... 59  
   A.2. Middlebury ......................................................... 62  
   A.3. Leuven ............................................................. 64  

B. **Statistics** ............................................................ 69  
   B.1. Middlebury ......................................................... 69  
   B.2. Leuven ............................................................. 74  
      B.2.1. Apical 2 chamber .......................................... 74  
      B.2.2. Apical 3 chamber .......................................... 76  
      B.2.3. Apical 4 chamber .......................................... 78  

**Bibliography** ............................................................. 81
Medical ultrasound is non-invasive image modality used in diagnostic applications. It has lower cost than other methods, such as MR and CT, and does not use any harmful ionizing radiation. Some of the drawbacks are limitations on the field of view and problems with structures such as bone and air, e.g. lungs can not effectively be imaged by ultrasound (Szabo 2013). It has become the preferred modality in cardiology since it gives real time images, and uses portable machines that can be brought bedside.

The coherent nature of ultrasound results in images with speckle noise that reduced its utility for less than highly trained users and also complicates image processing tasks such as feature segmentation (Krissian et al. 2007). The structure tensor (Knutsson 1989) can be used for estimating the direction of the structure in the ultrasound image. This estimate can then be used in oriented filtering for speckle reduction (Krissian et al. 2007).

Cardiac Strain evaluates the function of the heart by measuring the deformation of the heart muscle. In one dimension, an object can only shorten or lengthen, and the instantaneous strain is defined as (D’hooge, Heimdal, et al. 2000):

$$\varepsilon(t) = \frac{L(t) - L(t_0)}{L(t_0)}$$

where $L(t_0)$ is an initial length measurement, and $L(t)$ is the length at time $t$. A two dimensional object can not only lengthen and shorten along the axes (normal strain), but also distort in shape (shear strain). Two dimensional strain can be estimated by speckle tracking (D’hooge, Bijnens, et al. 2002).

Historically speckle tracking has been performed with either block matching techniques such as correlation (Friemel, Bohs, and Trahey 1995), or optical flow techniques (Lucas and Kanade 1981; Suhling et al. 2003). A dense estimation of the heart motion is a costly operation, therefore a segmentation of the image is usually done to define region of interest. A survey of segmentation methods was done by Noble and Boukerroui (2006), wherein they reported that techniques based on the analytic signal (Gabor 1946) perform well. Recently, Alessandrini, Basarab, et al. (2013) proposed the use of the monogenic signal (Felsberg and Sommer 2001b), which is an extension of the analytic signal to higher dimensions, for estimating heart motion. Speckle tracking is also used for 2D blood velocity estimation (Fadnes et al. 2015).
1. Introduction

Problem statement
What is the performance impact of replacing the gradient in optical flow methods with either a monogenic signal (Felsberg and Sommer 2001b) based method or the quadrature structure tensor (Knutsson 1989) on optical images and echocardiograms?
2 Background

2.1. Optical flow

2.1.1. Introduction

One of the fundamental problems in processing image sequences is estimating the image motion. This consists of moving, and possibly deforming, an image so that the difference to the image we are aligning to is minimized. The estimated image motion is often referred to as optical flow (Horn and Schunck 1981). Once the optical flow is computed, it can be used in several computer vision problems, such as stereo vision, video stabilization, tracking and motion interpolation.

The objective of stereo vision is to obtain 3D information from a scene captured with two cameras. The need for two cameras comes from the fact that an image is a 2D projection of the 3D world, so a single image is not enough to reconstruct the data-points in 3D (Lucas and Kanade 1981). Since every point in space forms a triangle between the two cameras and that point, the optical flow can be used to calculate the opening angle at the top of the triangle. If the angle is small, the object is far away, and if the angle is large, the object is close, see Figure 2.1 for a simple visualization. This information can then be used for tasks such as navigation, mapping and collision avoidance (Szeliski 2011).

The goal of a video stabilization algorithm is to eliminate high frequency oscillation, typically due to undesired camera shake, often referred to as jitter, from the video. This is achieved by applying a corrective motion onto each video frame, such that the undesired motion is removed from the output video. The corrective motion is calculated as the difference from the global motion and the intended motion. In the case of a stationary camera, e.g. a traffic camera with motion induced by wind or traffic vibration, the intended motion is zero. One approach is to select a patch with small accumulated motion, i.e. a part of the background in the image, and use the optical flow in this path as the corrective motion. If the camera is also moving through space, e.g. mounted on a drone or a car, the global motion contains both the intended and undesired motion. Usually an assumption that the intended motion is smooth is used, such that any non-smoothness in the global motion can be attributed to the undesired motion. If the
2. Background

Figure 2.1.: Simplified depth perception using stereo vision. The object farther away has a smaller angle ($\theta_2$) than the object closer to the camera pair ($\theta_1$).

Figure 2.2.: Example of interpolating along a path at an unknown point, $t + \frac{1}{2}$, between two known points $t$ and $t + 1$.

Jitter is large, the output video usually contains visual artifacts such as black borders. See the paper by Matsushita et al. (2006) for an overview of a full video stabilization pipeline.

In tracking applications we want to follow an object through several images, usually representing a time series. Optical flow is used to draw the path of the object through the image series. If the image series has a lot of jitter, it may be necessary to stabilize before tracking. This has application in a large range of fields, such as traffic surveillance, sports television and medical diagnosis (Yamane, Shirai, and Miura 1998). For example, a sports presenter may want to show how soccer players moved on the pitch during an attack.

Motion interpolation is used to increase the frame rate of a video. This is often done as part of the input pipeline of televisions in order to match the video frame rate to the refresh rate of the LCD panel. The aim is to fill in frames inbetween existing frames, often with the objective of making the perceived video more smooth. The optical flow is used to draw a path between frames, and the interpolated frames is drawn on the path (Mahajan et al. 2009), see Figure 2.2.
2.1. Optical flow

**Brightness constancy**

In order to locate the same object in both images, we make an assumption that the object does not change between the two images. This is called the **brightness constancy constraint** (Horn and Schunck 1981), referred to by the shorthand BCCE, defined as

\[ g(x, y, t) = g(x + u, y + v, t + \Delta t), \]  

(2.1)

where \( g \) is the grayscale image intensity at point \((x, y)\) at time \( t \), \((u, v)\) is the motion of the object located at \((x, y)\), and \( \Delta t \) is the temporal sampling interval. In real world images this constraint seldom holds, as the perceived color of an object is dependent on several variables outside of the object, such as lighting changes and occlusions.

Horn and Schunck (1981) used a Taylor expansion on the right hand side of (2.1), and reformulated it as

\[ (\nabla_x g)^T u = -\nabla_t g, \]  

(2.2)

where \( \nabla_x g = (\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}) \) and \( \nabla_t g \) is the spatial and temporal gradients of the image intensity and \( u = (u, v) \) respectively. This is an under constrained problem, since there are two unknowns, but only one equation. To fully recover \( u \), extra constraints are needed.

**Independent motion estimation at each pixel**

When doing simple image registration, e.g. when estimating camera movement in the same plane as non-moving objects, only a single motion vector is needed. If the camera movement is more complex, or the objects are moving, more motion vectors are needed. Thus the brightness constancy constraint in (2.1) can be reformulated to be dependent on \( x \):

\[ g(x, t) = g(x + u(x), t + \Delta t). \]  

(2.3)

For simplicity, when the dependency on \( x \) is not explicitly needed, \( u(x) \) will be written as \( u \).

**Aperture problem**

When the image does not capture the whole object for which we are estimating the motion, ambiguities arise. This is called the **aperture problem**. Figure 2.3 shows a line moving through a circular aperture. Since the edge of the line is not captured, a large set of motions could explain the motion of the line. This can also be seen in the well known “barber pole” illusion, where the perceived motion is upwards, but the real motion is sideways, as illustrated in Figure 2.6.

**Pyramids**

One way to make the subspace of possible motions smaller, is to low pass or band pass filter the original image. Since low pass filtered images can be downsampled without loss of information
2. Background

Figure 2.3.: The motion between the two lines is ambiguous inside the circular aperture. The arrows show three different possible motions.

Figure 2.4.: Example of motion with distributed spatial structure (moving corner)

Figure 2.5.: Image pyramid

Figure 2.6.: The barber pole rotates around the long axis, but the perceived motion of the colored lines is upwards.
2.1. Optical flow

(Oppenheim and Schafer 2010), the motion that is retained in the downsampled version, covers larger motion over fewer pixels. This approach is referred to as coarse-to-fine image pyramids. The n-th layer of the pyramid is referred to as $g^{(n)}$, where the bottom layer $g^{(0)}$ is the original image. Adelson et al. (1984) proposes to use small and separable Gaussian-like filter kernel to generate the image pyramid. Figure 2.5 illustrates how to iteratively create an image pyramid using a Gaussian-like filter kernel.

Using the motion on layer $n$ as a constraint on the motion for layer $n - 1$ was proposed by (Lucas and Kanade 1981). Anandan (1989) used the same sized region in pixels for all layers, but uses the estimation on layer $n$ as the center for the search region in layer $n - 1$. One problem with the pyramid approach is that small objects with large movement is not captured in the higher layers, and the movement may be missed completely in the lower layers (Brox and Malik 2011).

Search whole displacement space

The obvious, and somewhat naïve approach to finding $u$, is to calculate the error at all possible motions, i.e. to search the whole subspace of possible motions. Even for a modest number of possible motions, this becomes very costly. If the size of the image is $N \times N$, and the size of the subspace of possible $u$ is $M \times M$, the computation complexity is $O(M^2N^2)$ (Lucas and Kanade 1981). Algorithms with lower computation complexity will be discussed in later sections.

Interpolation

Given non-integer motion estimates, it is necessary to interpolate $g(x + u, t + \Delta t)$. This is usually done by piecewise polynomials, better known as splines. Splines of degree 1 are equivalent to a piecewise linear interpolation. For better interpolation a spline of degree 2 can be used. Unser, Aldroubi, and Eden (1991) shows a fast cubic spline interpolation algorithm which uses recursive FIR-filters, and also describes how to compute the cascade filters for higher order interpolation.

Dense vs sparse flow

If the image is large, estimating motion at all points can be quite time consuming. Therefore, some algorithm for finding good features in the image, often called keypoints, is needed. This is usually points in the image with high contrast or distributed spatial structure, such as corners and edges, see Figure 2.4 for one example. Several algorithms have been proposed, such as the Shi-Tomasi corner detector by Shi and Tomasi (1994) and SIFT (Scale-Invariant Feature Transform) by Lowe (2004).
2. Background

**Motion models**

The model of the motion between two images can take several forms. The simplest form is a translation. Formulated as:

\[
x + u(x) = \begin{bmatrix} x_1 + u \\ x_2 + v \end{bmatrix},
\]

where the vector \( u \) is then the optical flow at \( x \). If the object is moving in 3D space, a more complex model, such as an affine transform may be necessary:

\[
W(x; p) = \begin{bmatrix} p_1x_1 + p_3x_2 + p_5 \\ p_2x_1 + p_4x_2 + p_6 \end{bmatrix} = \begin{bmatrix} p_1 & p_3 \\ p_2 & p_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} p_5 \\ p_6 \end{bmatrix} = Ax + b,
\]

where \( p \) is the parameter vector for the affine transform, \( b \) represents the translation, and \( A \) is the four elementary geometric transforms: rotation, shear, dilation and stretching. With some abuse of notation, we can write

\[
W(x; p, t + n\Delta t) = g(W(x; p), t + n\Delta t) = g_u(W(x; p)).
\]

This is often referred to as the warped image. The flow \( u \) can then be calculated by

\[
u(x; p) = W(x; p) - x.
\]

In general \( W \) can be arbitrarily complex, leading to arbitrarily large \( p \).

**Selection of error metric.**

In order to calculate the error in the estimated motion, a metric for the difference between \( g_0(x) \) and \( g_1(x + u) \) is needed. Some common metrics are (Szeliski 2011):

- **\( \ell_1 \)-norm:**
  \[
  \sum_x |g_1(x + u) - g_0(x)|
  \]
  (2.8).

- **Sum of squared differences (SSD):**
  \[
  \sum_x |g_1(x + u) - g_0(x)|^2
  \]
  (2.9).

- **Normalized cross-correlation:**
  \[
  \frac{\sum_x (g_0(x) - \bar{g})(g_1(x + u) - \bar{g})}{\left(\sum_x (g_0(x) - \bar{g})^2\right)^{1/2} \left(\sum_x (g_1(x + u) - \bar{g})^2\right)^{1/2}}
  \]
  (2.10),

where \( \bar{g} = \frac{1}{MN} \sum_x g(x) \) for a \( M \times N \) image, i.e. the mean image.

2.1.2. Lucas Kanade

The brightness constancy constraint (2.2) is only looking at one pixel at a time. In a translation motion model, this gives 2 unknowns, but only one equation. Lucas and Kanade (1981) uses a region \( R \) of size \( n \times n \), around the point \( x \), and adds the constraint that the motion \( u \) is constant inside the region. This gives \( n^2 \) equations in 2 unknowns. Given a large enough region, the estimated optical flow is unlikely to be ambiguous (Horn 1988).
2.1. Optical flow

Error metric

Lucas and Kanade used the SSD (2.9) as the error metric for the estimated flow, and Taylor expanded the warped image. The parameter vector is iteratively updated, and the error for the n-th iteration then becomes:

\[
E_n = \sum_{x \in \mathbb{R}} (g_1(W(x; p_{n-1} + \Delta p_n)) - g_0(x))^2
\]

\[
= \sum_{x \in \mathbb{R}} \left( g_1(W(x; p_{n-1})) + \nabla_x g_1(W(x; p_{n-1})) \frac{\partial W}{\partial p}(x) \Delta p_n - g_0(x) \right)^2.
\]  

(2.11)

Where \( \Delta p_n \) is the n-th parameter update, \( \frac{\partial W}{\partial p}(x) \) is the Jacobian of the warp function evaluated at \( x \), and \( \nabla_x g_1(W(x; p_{n-1})) \) is the spatial image gradient evaluated at \( W(x; p_n) \); i.e. the image gradient is calculated in the original reference frame, and then evaluated at the warped reference frame.

Minimization

In order to minimize the error \( E_n \) with respect to the parameter update \( \Delta p_n \), the derivative of the error with respect to the update is set to zero

\[
0 = \nabla_{\Delta p_n}(E_n)
\]

\[
= \sum_{x \in \mathbb{R}} \nabla_x g_1(W(x; p_{n-1}))^\top \left( \nabla_x g_1(W(x; p_{n-1})) \frac{\partial W}{\partial p}(x) \Delta p_n + 2 (g_1(W(x; p_{n-1})) - g_0(x)) \right).
\]

(2.12)

The update then becomes

\[
\Delta p_n = \left( \sum_{x \in \mathbb{R}} \nabla_x g_1(W(x; p_{n-1}))^\top \nabla_x g_1(W(x; p_{n-1})) \right)^{-1} \times \sum_{x \in \mathbb{R}} \left( \nabla_x g_1(W(x; p_{n-1})) \frac{\partial W}{\partial p}(x) \right)^\top 2 (g_1(W(x; p_{n-1})) - g_0(x)).
\]

(2.13)

The first part of the update equation is the Gauss-Newton approximation to the Hessian matrix

\[
H = \sum_{x \in \mathbb{R}} \nabla_x g_1(W(x; p_{n-1}))^\top \nabla_x g_1(W(x; p_{n-1})) \approx \begin{bmatrix}
\frac{\partial^2 g_1}{\partial x_1^2} & \frac{\partial^2 g_1}{\partial x_1 \partial x_2}\\
\frac{\partial^2 g_1}{\partial x_2 \partial x_1} & \frac{\partial^2 g_1}{\partial x_2^2}
\end{bmatrix}
\]

(2.14)

As long as the double derivatives of \( g_1 \) are small, the error of the linearization by Taylor expansion is also small. The multiplication by the inverse of the Hessian can be seen as a windowing, based on how close the Taylor approximation is to the original \( g_1 \).

The second part of (2.13) is an estimation of the time derivative.
2. Background

Iterative scheme

Let \(e_n(x) = g_1(W(x;p_n)) - g_0(x)\) denote the warp error at \(x\) given \(p_n\). The iterative scheme then becomes

\[
p_{n+1} = p_n + \Delta p_{n+1} = p_n + H^{-1} \sum_{x \in \mathcal{R}} \left( \nabla_x g_1(W(x;p_n)) \frac{\partial W}{\partial p}(x) \right)^T e_n(x).
\]  
(2.15)

A normal stopping criterion for the iterative scheme is when the norm of the update is below some threshold (Baker and Matthews 2002)

\[
\|\Delta p_n\| < \varepsilon. 
\]  
(2.16)

If the motion model used is not pure translation, the flow can then be calculated with (2.7).

2.1.3. Horn Schunk

Horn and Schunck (1981) suggested a constraint on the brightness constancy constraint that forces the flow field to be smooth, by assuming that close points in the image have similar velocities. They used the Frobenius norm of the Jacobian of the optical flow vector as the measure of departure from smoothness

\[
E_s(u(x)) = \|\nabla u(x)\|_F^2 = \left( \frac{\partial u}{\partial x}(x) \right)^2 + \left( \frac{\partial u}{\partial y}(x) \right)^2 + \left( \frac{\partial v}{\partial x}(x) \right)^2 + \left( \frac{\partial v}{\partial y}(x) \right)^2. 
\]  
(2.17)

Error metric

Horn and Schunck used the L2-norm of the gradient formulation of the color constancy (2.2) and added the smoothness constraint (2.17)

\[
E(u) = \int_{\mathcal{R}} \left( \left( \nabla_x g(x)^T u(x) + \nabla_y g(x) \right)^2 + \alpha^2 \|\nabla u(x)\|_F^2 \right) \, dx,
\]  
(2.18)

where \(\mathcal{R}\) usually is the whole image and \(\alpha\) controls the relative importance between the color constancy constraint and the smoothness constraint. A larger \(\alpha\) gives a smoother flow. This is often referred to as a global method due to the extent of \(\mathcal{R}\).

Minimization

In order to find the \(u\) that minimizes \(E(u)\), Horn and Schunck solved the associated Euler-Lagrange equations, and obtained

\[
(\nabla_x g)^2 u + \nabla_x g \nabla_y g v = \alpha^2 \nabla^2 u - \nabla_x g \nabla_t g, 
\]  
(2.19a)

\[
\nabla_x g \nabla_y g u + (\nabla_y g)^2 v = \alpha^2 \nabla^2 v - \nabla_y g \nabla_t g. 
\]  
(2.19b)
where $\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ and $\nabla^2 v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}$ are the Laplacian of $u$ and $v$, and $\nabla_x g$ and $\nabla_y g$ are the image gradients with respect to $x$ and $y$. The Laplacian is estimated by subtracting the value of $u(x)$ from the average of the neighborhood around $x$. The neighborhood average is denoted by $\hat{u}(x)$, and the approximation becomes

$$\nabla^2 u(x) \approx \frac{1}{h^2} (\hat{u}(x) - u(x)), \quad \nabla^2 v(x) \approx \frac{1}{h^2} (\hat{v}(x) - v(x)),$$

(2.20)

where $h$ is the step size of the grid, for unit step size it disappears. Horn and Schunck used a weighted 8-neighborhood to calculate $\hat{u}$ and $\hat{v}$

$$\hat{u} = u * H = u * \frac{1}{12} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 0 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$ 

(2.21)

By using (2.20) in (2.19a) and (2.19b) we get

$$\left((\nabla_x g)^2 + \alpha^2\right) u + \nabla_x g \nabla_y g u = \alpha^2 \hat{u} - \nabla_x g \nabla_t g,$$ 

(2.22a)

$$\nabla_x g \nabla_y g u + \left((\nabla_y g)^2 + \alpha^2\right) v = \alpha^2 \hat{v} - \nabla_y g \nabla_t g,$$ 

(2.22b)

which is linear in $u$ and $v$. Solving the set of linear equations for $u$ and $v$ gives

$$\left(\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2\right) u = \left(\alpha^2 + (\nabla_y g)^2\right) \hat{u} - \nabla_x g \nabla_y g \hat{v} - \nabla_x g \nabla_t g,$$ 

(2.23a)

$$\left(\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2\right) v = -\nabla_x g \nabla_y g \hat{u} + \left(\alpha^2 + (\nabla_y g)^2\right) \hat{v} - \nabla_x g \nabla_t g.$$ 

(2.23b)

**Iterative scheme**

Solving the equations (2.23a) and (2.23b) with a direct method, such as Gauss-Jordan elimination, would be costly since the corresponding matrix is large and sparse. By rewriting (2.23a) and (2.23b) we get

$$\left(\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2\right) (u - \hat{u}) = -\nabla_x g \left(\nabla_x g \hat{u} + \nabla_y g \hat{v} + \nabla_t g\right),$$ 

(2.24a)

$$\left(\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2\right) (v - \hat{v}) = -\nabla_y g \left(\nabla_x g \hat{u} + \nabla_y g \hat{v} + \nabla_t g\right).$$ 

(2.24b)

which leads to an iterative scheme where the updated $u_{n+1} = (u_{n+1}, v_{n+1})^T$ is computed from an average of the previous flow estimate $u_n$ and the estimated derivatives

$$u_{n+1} = \hat{u}_n - \nabla_x g \frac{\nabla_x g \hat{u} + \nabla_y g \hat{v} + \nabla_t g}{\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2},$$ 

(2.25a)

$$v_{n+1} = \hat{v}_n - \nabla_y g \frac{\nabla_x g \hat{u} + \nabla_y g \hat{v} + \nabla_t g}{\alpha^2 + (\nabla_x g)^2 + (\nabla_y g)^2}.$$ 

(2.25b)
2. Background

If the image gradients is zero, i.e. the image region is uniform, the updated flow estimates \( u_{n+1} \) will simply be an average of the previous neighboring flow estimates. Eventually, the whole uniform region in the image has been filled with a flow estimate. If the flow on the border are all equal to the same value, that value will fill the whole region. If the flow on the border varies, the values on the interior of the region will correspond to the solutions of the Laplace equation for the given boundary conditions (Tveito and Winther 2005).

The in-filling phenomena gives us an lower bound for the number of iterations required to have a dense flow. It must be larger than the number of pixels across the largest uniform region in the picture. If this is not known in advance, the diagonal length of the image may be used as a conservative estimate.

Another stopping criteria is when the update is below a threshold \( \varepsilon \), similar to the stopping criteria (2.16) in the Lucas Kanade method

\[
\|u_{n+1} - u_n\| < \varepsilon. \tag{2.26}
\]

2.1.4. Limitations

Size of region in Lucas-Kanade

The selection of the region size for \( R \) in Lucas-Kanade leads to a dilemma. We want the size of \( R \) to be large in order for it to be noise insensitive and sufficiently constrain the solutions. The problem is that when the region grows, the likelihood of it containing multiple motions increases, i.e. our assumption on the motion no longer holds. When there is multiple motions in the region \( R \), no single \( u \) will give a small error over there whole region. This results in a solution from the iterative scheme (2.15) that lies somewhere between the two true motions in the \( u,v \)-space (M. J. Black and Anandan 1996). Thus we want the region to be small in order to avoid error in the estimates. This dilemma is referred to as the generalized aperture problem (Jepson and Michael J. Black 1993).

Motion boundaries

Motion boundaries is a problem for both algorithms presented, since it violates the prior assumption on the motion. The constraint added by Horn and Schunck leads to over smoothing of the estimated flow (M. J. Black and Anandan 1996). In Lucas-Kanade this leads to a solution that lies somewhere between the two true motions. Okutomi and Kanade (1992) proposed an adaptive size of the region \( R \) in order to minimize the uncertainty in the estimate.

Lack of confidence measure

None of the above algorithms include a confidence measure in the original papers. The purpose of a confidence measure is to indicate the reliability of the computed flow (Beauchemin and Barron 1995). Simoncelli, Adelson, and Heeger (1991) presented a Bayesian view on finding
2.2. Space-time images

the optical flow with constant motion inside a region. This lead them to propose a confidence measure based on the eigenvalues of the approximated Hessian matrix. Simoncelli, Adelson, and Heeger used the sum of the eigenvalues, but Barron, Fleet, and Beauchemin (1994) found that thresholding each eigenvalue individually gives a better result. Anandan (1989) proposed a confidence measure based on the curvature of the error surface.

Sensitivity to noise

A common problem with gradient methods is that the gradient operator “amplifies” high frequency noise (Bertero, Poggio, and Torre 1988). Let the function $f(x)$ be perturbed by a single frequency such that $g(x) = f(x) + \varepsilon \sin(\omega x)$. The difference between $g(x)$ and $f(x)$ can be arbitrarily small given very small $\varepsilon$. But $g'(x) = f'(x) + \varepsilon \omega \cos(\omega x)$ may be very different from $f'(x)$ given a large enough $\omega$. By low pass filtering the images we can establish an upper bound on the noise introduced by the gradient operation.

2.2. Space-time images

The BCCE (2.2) as constructed above is restricted to two consecutive images. The notion of a grayscale image can be extended into a continuous spatio-temporal image, also known as the $xt$-space (Jähne 1993). This extension can be thought of as a stack of consecutive images forming a space-time cube. Each pixel is then extended to a voxel spanning a small volume in the space-time cube.
2. Background

**Motion analysis**

Analysing the motion in a space time image with one spatial dimension reveals that different motions yields different structures, see Figure 2.7a. The velocity is directly connected to the orientation of the structure

\[ u = -\tan \varphi, \quad (2.27) \]

where \( \varphi \) is the angle between the t-axis and the direction of constant gray value. The minus sign is because angles are positive counterclockwise. Extending (2.27) to 2D is straight forward, see Figure 2.7b

\[ u = -\begin{bmatrix} \tan \varphi_x \\ \tan \varphi_y \end{bmatrix}, \quad (2.28) \]

where \( \varphi_x \) and \( \varphi_y \) is the angle between the t-axis and the x and y components of a vector pointing in the direction of constant gray values. Thus the search for motion reduces to an orientation analysis in the \( xt \)-space.

With the insight that we are looking for the direction of the constant gray value in the spatio-temporal image, the BCCE can be reformulated

\[ (\nabla_{xt} g)^\top \mathbf{v} = 0, \quad (2.29) \]

where \( \mathbf{v} \) is the direction of constant gray value, and thus the gradient vanishes in this direction.
3 Monogenic Signal

3.1 1D signal and the analytic signal

The analytic signal (Gabor 1946) is a much used technique in signal processing for analyzing 1D-signals. It is an extension of a real valued signal into a complex valued signal by the Hilbert transform.

3.1.1 The Hilbert transform

The Hilbert transform in the time domain is a singular integral

\[ \mathcal{H}(g(t)) = \frac{1}{\pi} \text{p.v.} \int_{-\infty}^{\infty} \frac{g(\tau)}{t - \tau} \, d\tau, \quad (3.1) \]

where p. v. means that the integral should be taken as a Cauchy principal value because of the singularity at \( t \). It is defined as

\[ \text{p. v.} \int_{-\infty}^{\infty} g(\tau) \, d\tau = \lim_{\epsilon \to 0} \left( \int_{t-\epsilon}^{t} g(\tau) \, d\tau + \int_{t}^{t+\epsilon} g(\tau) \, d\tau \right). \quad (3.2) \]

Thus the Hilbert transform can be seen as a convolution

\[ \mathcal{H}(g(t)) = \left( \frac{1}{\pi t} * g \right)(t) = (K_H * g)(t). \quad (3.3) \]

In the Fourier domain this gives

\[ (\mathcal{K}_H * g)(\omega) = \mathcal{K}_H(\omega) \mathcal{F}(\omega) = (-j \text{sgn} \omega) \hat{g}(\omega) = -j \frac{\omega}{|\omega|} \hat{g}(\omega), \quad (3.4) \]

where \( j = \sqrt{-1} \). Thus the Hilbert transform flips the sign of the negative frequencies and introduces a \(-\frac{\pi}{2}\) phase-shift. See Figure 3.1 for an example. If the signal \( g(t) \) is real, it has a symmetric Fourier spectrum, and the spectrum of the Hilbert transformed signal is then anti-symmetric.
3. Monogenic Signal

Figure 3.1.: Example showing the calculation of the analytic signal in the Fourier domain. (a): The spectrum of a signal $g(t)$. (b): Hilbert-transform of $g(t)$. (c): The analytic signal formed as the sum of (a) and (b).

### 3.1.2. The analytic signal

Given a real signal $g(t)$, the analytic signal, $g_a(t)$, is given by

$$g_a(t) = g(t) + j \hat{g}(t) \quad \rightarrow \quad \hat{g}_a(\omega) = \hat{g}(\omega) + \frac{\omega}{|\omega|} \hat{g}(\omega). \quad (3.5)$$

The analytic signal is complex valued since it’s spectrum is single-sided, and thus it has a modulus and argument. This is referred to as instantaneous phase and instantaneous amplitude, or more simply, just phase and envelope

$$A(t) = |g_a(t)|, \quad (3.6)$$

$$\phi(t) = \text{arg}(g_a(t)). \quad (3.7)$$

This is called a split of identity (Ville 1948), and the analytic signal can be written in polar form

$$g_a(t) = A(t)e^{-j\phi(t)}. \quad (3.8)$$

See Figure 3.2 for an example of envelope and phase of a signal.

### 3.1.3. Properties of the analytic signal

The Hilbert transform and the analytic signal has several important properties (Felsberg and Sommer 2001b):

- The transfer function is anti-symmetric: $\hat{K}_H(-\omega) = -\hat{K}_H(\omega)$.
- It suppresses the DC energy: $\hat{K}_H(0) = 0$.
- Kernel has unit energy: $|\hat{K}_H(\omega)| = 1, \forall \omega \neq 0$.

This gives rise to the following properties of the analytic signal:
3.2. nD signals

Signals of higher dimensions, such as images (2D) and video (3D), contain information not easily captured by a 1D signal. Therefore a framework for characterizing 2D and 3D signals is needed.

3.2.1. Intrinsic dimensionality

The number of free parameters needed to describe a signal in a neighborhood is an important underlying property of a signal. This property is called local intrinsic dimensionality (Bennett 1969; Krieger and Zetzsche 1996), and by shorthand referred to as imD signal, where m is the intrinsic dimensionality. The word intrinsic is used in order to distinguish between the signal domain and the variation in the signal. A local i1D neighborhood is also known as a simple neighborhood (Granlund and Knutsson 1995). An example of a i1D signal, $g(x) = \sin(\omega(x + y/2))$, is shown in Figure 3.3b.

Formal definition

If a signal is imD, there exists a function $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$, and m functions $u_i : \mathbb{R}^n \rightarrow \mathbb{R}$, such that

$$
g(x) = f(u_1(x), \ldots, u_m(x)), \quad \forall x \in \mathcal{R},
$$

(3.9)

where m is the minimum number of functions that fulfills (3.9). This property is also called rank decomposition (Trefethen and Bau III 1997), and therefore the intrinsic dimension of a signal is equal to its rank.

Figure 3.2.: Example of envelope and phase of a windowed signal. (a): A windowed sinusoid. (b): Magnitude of the analytic signal. (c): The local phase of the analytic signal.

- It has two times the energy, since $f(t)$ and $j\tilde{H}(t)$ are orthogonal.
- Provides a split of identity, see (3.8).
- The spectrum is one sided: $\tilde{g}_a(\omega) = 0$, $\forall \omega < 0$. 
3. Monogenic Signal

![Figure 3.3.: Example of three 2D signal with different intrinsic dimensionality.](image)

**Fourier**

A signal that is not of full rank, must by definition be constant along one or several dimensions. Since the Fourier transform of a constant is the impulse function $\delta(\omega)$, the spectrum of a signal with intrinsic dimensions $m < n$, is the impulse function along the constant dimensions.

**Real signals**

As real world measured signals contains noise, and most noise is 2D (Felsberg, Kalkan, and Krüger 2009), the strict definition (3.9) is not useful for determining the intrinsic dimension. One way to resolve this problem is to add an error term, $E$, to (3.9)

$$g(x) = f(u_1(x), \ldots, u_m(x)) + E(x), \quad \forall x \in \mathbb{R}$$

(3.10)

This is similar to a low rank matrix approximation (Eckart and Young 1936), and the intrinsic dimension can be approximated by the number of eigenvalues larger than a threshold.

**3.3. Extension of the analytic signal to higher dimensions**

Looking at (3.5), a naïve extension of the analytic signal to higher dimension could be

$$g_a(x) = g(x) + j\mathcal{H}g(x).$$

(3.11)

The 1D analytic signal is construed by suppressing negative frequencies, but in higher dimensions the definition of what a negative frequency is is not clear, and thus the definition of $\mathcal{H}(f(x))$ in (3.11) is not straightforward. When designing a higher dimensional extension of the analytic signal, the properties laid out in subsection 3.1.3 must be preserved as much as possible. There have been several proposals for such an extension. The most notable examples are the total Hilbert transform (Stark 1971), partial Hilbert transform (Granlund and Knutsson 1995), and the Riesz transform (Felsberg and Sommer 2001b).
3.3. Extension of the analytic signal to higher dimensions

3.3.1. Total Hilbert transform

In the total Hilbert transform the signum function is taken as a product of all the components (Stark 1971)

$$\mathcal{H}_T g(\omega) = \left( -j \right)^n \prod_{i=1}^{n} \text{sgn}(\omega_i) \hat{g}(\omega). \quad (3.12)$$

and the total analytic signal is given by

$$g_\omega(x) = g(x) + j\mathcal{H}_T g(x). \quad (3.13)$$

The spectrum of (3.13) is present in every quadrant of the spectrum (Bulow 1999). Thus there is no correspondence to the one-sidedness of the 1D analytic signal. Orthogonality between $g(x)$ and $j\mathcal{H}_T g(x)$ is only present if $g$ is separable, i.e. $g(x) = \prod g(x_i)$. Thus the total analytic signal is not a valid generalization of the analytic signal.

3.3.2. Partial Hilbert transform

The idea behind the partial Hilbert transform is to take signum function to identify on which side of a hyperplane the frequency vector lies. Given a unit vector $e$ that is normal to the hyperplane, the extended signum function is defined as

$$\text{sgn}_e(\omega) := \begin{cases} 
1 & \text{if } \omega^T e > 0 \\
0 & \text{if } \omega^T e = 0 \\
-1 & \text{if } \omega^T e < 0 
\end{cases} \quad (3.14)$$

The partial Hilbert transform is then defined in the Fourier space as (Granlund and Knutsson 1995)

$$\mathcal{H}_P g(\omega) = (-j \text{sgn}_e(\omega))\hat{g}(\omega), \quad (3.15)$$

and the partial analytic signal is given by

$$g_\omega(x) = g(x) + j\mathcal{H}_P g(x). \quad (3.16)$$

If the signal $f(x)$ is simple, i.e. 1D, then it has a clear direction, and this direction can be used in the partial Hilbert transform. The main drawback by this approach is that $\mathcal{H}_P$ has a preferred direction. It is thus not isotropic and therefore not a valid generalization of the analytic signal.
3. Monogenic Signal

3.3.3. Riesz transform

Instead of defining a transform that has the same domain and range as the signal, another approach is to keep the dimensionality of the domain. The Riesz transform is one such approach (Felsberg and Sommer 2001b)

\[ \hat{R_R}(\omega) = -j \frac{\omega}{\| \omega \|} \hat{g}(\omega) = \hat{K_R}(\omega)\hat{g}(\omega). \]  \hspace{1cm} (3.17)

It can be separated (Stein 1970)

\[ R_R(x) = [R_1g(x) \cdots R_ng(x)]^T, \]  \hspace{1cm} (3.18)

with the i-th component given by

\[ \hat{R_iR}(\omega) = -j \frac{\omega_i}{\| \omega \|} \hat{g}(\omega). \]  \hspace{1cm} (3.19)

Hence the Riesz transform can be written as a convolution

\[ R_i(\omega) = (K_{R_i} \ast g)(x). \]  \hspace{1cm} (3.20)

In the time domain it becomes a singular integral (Stein 1970)

\[ R_i(x) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\pi^{(n+1)/2}} \lim_{\varepsilon \to 0} \int_{\mathbb{R}^n \setminus B_\varepsilon(x)} \frac{x_i - y_i}{\| x - y \|^{n+1}} g(y) \, dy, \]  \hspace{1cm} (3.21)

where \( \Gamma(n) = (n-1)! \) is the Euler gamma function and \( B_\varepsilon(x) \) is the ball of radius \( \varepsilon \) around \( x \).

Thus \( R_i \) is defined by the kernel

\[ K_{R_i}(x) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\pi^{(n+1)/2} \| x \|^{(n+1)}} \frac{x_i}{\| x \|^{(n+1)}}, \]  \hspace{1cm} (3.22)

where \( c_n \) is a normalization constant. In 1D \( c_n \) becomes \( 1/\pi \) and thus \( K_{R_i} = K_H \) for 1D.

The Riesz transform has the following properties:

- The transfer function is anti-symmetric at points: \( \hat{K_R}(-\omega) = -\hat{K_R}(\omega) \).
- Suppresses the DC energy: \( \hat{K_R}(0) = 0 \) (Felsberg and Sommer 2001b).
- Has unit norm in every direction, i.e. isotropic \( \| \hat{K_R}(\omega) \| = 1, \forall \omega \neq 0 \).
3.4. Monogenic signal

Connection with the gradient operator

The partial derivative has the following Fourier transform pairs

\[
\frac{\partial f(x)}{\partial x_i} \overset{\mathcal{F}}{\leftrightarrow} -j\omega_i \hat{g}(\omega).
\] (3.23)

Comparing the right side of (3.23) with (3.19) there is a clear resemblance, but the \(|\omega|^{-1}\) is not accounted for in (3.23). Therefore another operator is needed. The Laplace-operator is given by

\[
\Delta = \text{div} \nabla = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2},
\] (3.24)

and its Fourier transform is

\[
\hat{\Delta}g(\omega) = -|\omega|^2 \hat{g}(\omega).
\] (3.25)

From this we can define the fractional Laplacian (Stein 1970) for \(\alpha \in (-n, n)\)

\[
(-\Delta^{\frac{\alpha}{2}} g)(\omega) = |\omega|^\alpha \hat{g}(\omega).
\] (3.26)

By combining (3.23) and (3.26) with \(\alpha = -1\) we get

\[
\mathcal{F}\left( \frac{\partial}{\partial x_i} \left( \frac{\partial}{\Delta^{\frac{1}{2}}} g \right) \right)(\omega) = -j\omega_i |\omega|^{-1} \hat{g}(\omega) = \hat{R}_i g(\omega).
\] (3.27)

Thus each component of the Riesz transform can be expressed as a partial derivative of the integral operator \(\Delta^{-\frac{1}{2}}\), which is an isotropic smoothing filter (Chenouard and Unser 2012)

\[
\hat{R}_i g = \frac{\partial}{\partial x_i} \left( -\Delta^{-\frac{1}{2}} g \right).
\] (3.28)

Therefore the Riesz transform should extract the same information as the gradient, but with less amplification of high frequencies.

3.4. Monogenic signal

Felsberg and Sommer (2001b) proposed the use of the Riesz transform in an extended analytic signal called the monogenic signal. Given an orthonormal basis \(\{e_i\}_{i=0}^{n}\), the monogenic signal can be expressed as

\[
g_m(x) = e_0 g(x) + \sum_{i=1}^{n} e_i \hat{R}_i g(x).
\] (3.29)
3. Monogenic Signal

In vector notation this becomes

\[ g_m(x) = \begin{bmatrix} g(x) & R_1g(x) & \cdots & R_ng(x) \end{bmatrix}^T = \begin{bmatrix} g(x) & g_1(x) & \cdots & g_n(x) \end{bmatrix}^T. \]  \tag{3.30}

Local features can then be extracted

\[ A(x) = \|g_m(x)\| = \sqrt{g^2(x) + g_1^2(x) + \cdots + g_n^2(x)}, \]  \tag{3.31}

\[ \phi(x) = \arg(g(x) + j\|Rg(x)\|), \]  \tag{3.32}

\[ \theta_i(x) = \frac{g_i(x)}{\|Rg(x)\|}, \quad i = 1, \ldots, n, \]  \tag{3.33}

where \( A(x) \) is the instantaneous amplitude, \( \phi(x) \) is the instantaneous phase, and \( \theta_i(x) \) is the \( i \)-th instantaneous orientation. In lower dimensions, the numbered subscripts is exchanged with names for the axis, e.g. in 3D

\[ g_m = \begin{bmatrix} g & g_x & g_y & g_t \end{bmatrix}^T. \]  \tag{3.34}

The monogenic signal has the following properties:

- It has two times the energy of the original signal, since \( \|\hat{R}g(\omega)\| = \|\hat{g}(\omega)\|, \quad \forall \omega \neq 0, \) and by construction \( g \) is orthogonal with every \( g_i, \quad i = 1, \ldots, n. \)

- Provides a split of identity into amplitude, phase and orientation.

- The spectrum is not one-sided, i.e. it contains redundant information, but this property is irrelevant for image recognition (Felsberg and Sommer 2001b). Nevertheless, there exist transforms such that the spectrum has a non redundant representation (Felsberg and Sommer 2001a).
As described in section 2.2, estimating flow between 2D images is the same as estimating orientation in the spatio-temporal image. This chapter presents a method for estimating the structure in 3D signals, and how to calculate the flow from the estimated structure. In section 4.1 the theory behind directional filtering is presented, followed by how to use it to estimate structure orientation in section 4.2. A method for calculating flow from the structure tensor is presented in section 4.3, and finally a connection between the structure tensor and Lucas-Kanade is shown in section 4.4.

4.1. Steerable filters

4.1.1. Introduction

The term *steerable filters* is used for filters which can synthesize a filter of arbitrary orientation as a linear combination of *basis filters* (Freeman and Adelson 1991). More formally the response at point \( x \) in direction \( \theta \) is defined as

\[
h(x; \theta) = \sum_{i=1}^{m} a_i(\theta) b_i(x),
\]

where \( a_i \) is the *interpolation function*, \( b_i \) are the *basis filters* and \( m \) is the number of filters.

4.1.2. Generalized directional derivatives

It is well known that the directional derivatives are steerable (Knutsson and Granlund 1983; Danielsson and Seger 1990; Farid and Simoncelli 1997). Simoncelli (1994) introduces the concept of inserting an arbitrary reversible pre-filter on the derivative. The derivative in direction \( e_\theta \) of a signal \( s \) is then given by

\[
\frac{\partial}{\partial e_\theta} (p * s)(x) = \left( \sum_{i=1}^{n} a_i(\theta) \left( \frac{\partial p}{\partial x_i} \right) * s \right)(x),
\]

(4.2)
4. Structure Tensor

(a) Vertical derivative $\frac{\partial G}{\partial x}$ (b) Horizontal derivative $\frac{\partial G}{\partial y}$ (c) $\cos(\frac{\pi}{3})\frac{\partial G}{\partial x} + \sin(\frac{\pi}{3})\frac{\partial G}{\partial y}$

Figure 4.1.: (a) and (b): The two derivatives of the Gaussian $G$. (c): A weighted sum of the derivatives gives a synthesized derivative in direction $\pi/3$.

and the basis filters are the partial derivatives of the pre-filter. The choice of pre-filter has been the subject of several papers (Simoncelli 1994; Farid and Simoncelli 1997; Scharr 2007). The interpolation function is dependent on the dimension of the domain. In 2 and 3 dimensions respectively we have

$$a(\theta) = \begin{bmatrix} \sin \theta & \cos \theta \end{bmatrix}^T, \quad a(\begin{bmatrix} \theta_1 & \theta_2 \end{bmatrix}) = \begin{bmatrix} \sin \theta_1 \cos \theta_2 & \sin \theta_1 \sin \theta_2 & \cos \theta_1 \end{bmatrix}^T.$$ (4.3)

**An example in 2D**

In order to give a feel for how the steerability properties work, an example is given for steerable derivatives. This example is inspired by the example given by Freeman and Adelson (1991).

Let $G$ be the 2D Gaussian function

$$G(x, y) = e^{-(x^2+y^2)},$$ (4.4)

and it’s derivatives are

$$b_0 = \frac{\partial G}{\partial x}(x, y) = -2xe^{-(x^2+y^2)} \quad b_1 = \frac{\partial G}{\partial y}(x, y) = -2ye^{-(x^2+y^2)}.$$ (4.5)

As seen in Figure 4.1a and b, $\frac{\partial G}{\partial x}$ and $\frac{\partial G}{\partial y}$ are copies of each other, with $\frac{\partial G}{\partial y}$ rotated by $\pi/2$. Thus they form a pair of basis filter $b_1$, and can be used to synthesize the derivative in any direction $\theta$. By using the 2D interpolator from (4.3), the derivative in direction $\theta$ of $s$ can be written as

$$d_\theta(x, y) = \cos \theta (b_0 * s)(x, y) + \sin \theta (b_1 * s)(x, y).$$ (4.6)

See Figure 4.1c for an example of the kernel steered in the direction $\pi/3$. Figure 4.2 shows the output of a synthetic image filtered with either of the Gaussian kernels, and the synthesized derivative in the direction of $\pi/3$. 

24
4.2. Structure tensor

4.2.1. Oriented signal energy

Applying the steerable filter (4.1) to a signal \( s(x) \) gives
\[
(h * s)(x; \theta) = \sum_{i=1}^{m} a_i(\theta) (b_i * s)(x).
\] (4.7)

The local energy around the point \( x \) in direction \( \theta \) is given by
\[
E(x, \theta) = \int_{\mathcal{R}} w(y) ((h * s)(x - y; \theta))^2 dy = \left\langle (h * s)(x; \theta)^2 \right\rangle_w,
\] (4.8)

where \( \langle f(x) \rangle_w \) is shorthand for smoothing with the local weight function \( w \) with support in the region \( \mathcal{R} \) around \( x \). By expanding the convolution and using \( g_i(x) = (b_i * s)(x) \) we get
\[
E(x, \theta) = \left\langle \left( \sum_{i=1}^{m} a_i(\theta) g_i(x) \right)^2 \right\rangle_w = \left\langle \left( \sum_{i=1}^{m} \sum_{k=1}^{m} a_i(\theta) a_k(\theta) g_i(x) g_k(x) \right) \right\rangle_w = A^T J(x) A \theta,
\] (4.9)

which is a quadratic form. The matrix \( J \) is known as the structure tensor (Knutsson 1989; Jähne 1993).

4.2.2. Gradient tensor

If the basis filters \( b_i \) in (4.1) are the partial derivatives, \( J \) is called the gradient tensor (Johansson and Farnebäck 2002). The number of filters is then the same as the dimension of the domain (Freeman and Adelson 1991), and the tensor components is given by
\[
J_{i,k}(x) = \left\langle \frac{\partial s}{\partial x_i}(x) \frac{\partial s}{\partial x_k}(x) \right\rangle_w.
\] (4.10)

Figure 4.2.: The derivatives of a circular image in three directions.
4. Structure Tensor

By (4.10) it is positive semi-definite since $J_{i,k} = J_{k,i}$. Then $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$ and the eigenvectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ forms an orthonormal basis for $\mathbb{R}^n$ (Trefethen and Bau III 1997).

4.2.3. Quadrature tensor

Knutsson (1989) introduces the use of polar separable filters in the Fourier domain for finding the structure tensor $J$. Polar separable means that the filter can be factored into a part $R$ depending on the radius, and one part $D$ dependent on the direction. The filter can then be calculated as the product

$$\hat{Q}(\omega) = R(\|\omega\|)D(\hat{\omega}),$$

(4.11)

where $\hat{\omega}$ is a unit vector in the same direction as $\omega$, and $\hat{Q} = \mathcal{F}(Q)$. Knutsson used the log-normal function for the radius part

$$R(\|\omega\|) = \exp\left(\left(\ln\|\omega\| - \ln \omega_0\right)^2 \frac{B}{2}\ln 2\right),$$

(4.12)

with $\omega_0$ the center frequency and $B$ the bandwidth. The directional part is

$$D(\hat{\omega}) = \left(\hat{\omega}^T d\right)^2 \text{sgn}(\hat{\omega}^T d),$$

(4.13)

where $d$ is a unit vector pointing in the filter direction. The directions is oriented in such a way that the filters are pairwise in quadrature. In the 3D case, Knutsson (1989) uses six directions

$$d_1 = c \begin{bmatrix} 1 & 0 \end{bmatrix}^T, \quad d_2 = c \begin{bmatrix} -1 & 0 \end{bmatrix}^T,$$

$$d_3 = c \begin{bmatrix} 0 & 1 \end{bmatrix}^T, \quad d_4 = c \begin{bmatrix} 0 & -1 \end{bmatrix}^T,$$

$$d_5 = c \begin{bmatrix} 0 & b \end{bmatrix}^T, \quad d_6 = c \begin{bmatrix} 0 & -b \end{bmatrix}^T,$$

(4.14)

with

$$a = 2, \quad b = \left(1 + \sqrt{5}\right), \quad \text{and} \quad c = \left(10 + 2\sqrt{5}\right)^{-1/2}. \quad (4.15)$$

The quadrature filters combine to create the structure tensor. The signal $s(x)$ filtered by the $i$-th quadrature filter is defined as

$$(Q_i * s)(x) = q_i(x).$$

(4.16)

The structure tensor $J$ can then be obtained by

$$J(x) = \sum_{i=1}^{m} |q_i(x)| \left(\alpha d_i d_i^T - \beta \mathbb{I}\right) = \sum_{i=1}^{m} |q_i(x)| M_i,$$

$$\alpha = \frac{5}{4}, \quad \beta = \frac{1}{4}. \quad (4.17)$$

Thus the structure tensor is a weighted sum of constant tensors $M_i$, corresponding to the $i$-th quadrature filter. The weights are the absolute value of the quadrature filter outputs. One drawback with the quadrature tensor is that it is guaranteed to be positive semi-definite only for simple signals (Johansson and Farnebäck 2002).
4.2.4. Riesz tensor

Chenouard and Unser (2011) proposed a formula for calculating the structure tensor from the components of the monogenic signal.

\[ J_{[i,k]}(x) = \langle R_i g(x) R_k g(x) \rangle_w. \]  

(4.18)

where \( i, k \in \{1, \ldots, n\} \). This is (4.10) with the partial derivatives exchanged with the \( i \)-th component of the Riesz transform, and it is positive semi-definite. Using the same notation as in the monogenic signal (3.34), the 3D tensor is given by

\[
J(x) = \begin{bmatrix}
\langle g_x^2(x) \rangle_w & \langle g_x(x) g_y(x) \rangle_w & \langle g_x(x) g_t(x) \rangle_w \\
\langle g_y(x) g_x(x) \rangle_w & \langle g_y^2(x) \rangle_w & \langle g_y(x) g_t(x) \rangle_w \\
\langle g_t(x) g_x(x) \rangle_w & \langle g_t(x) g_y(x) \rangle_w & \langle g_t^2(x) \rangle_w
\end{bmatrix}.
\]

4.3. Optical flow from the 3D structure tensor

As described in section 2.2, finding the optical flow is equivalent to finding the direction of constant gray value in a spatio-temporal image. In real world images subject to noise, there is not necessary a direction of constant gray value, but we can estimate the optical flow by finding the direction with least energy. Since a spatio-temporal image yields a 3D structure tensor, this direction is found by minimizing (4.9)

\[
\arg\min_{\theta} E(\theta) = a^T_\theta J a_\theta.
\]  

(4.19)

With the following interpolating function from (4.3). Optimizing with regards to \( \theta \) or \( a_\theta \) produces the same result, given the constraint that \( \|a_\theta\|^2 = a^T_\theta a_\theta = 1 \).

Optimization is carried out by the use of Lagrange multipliers, minimizing the Lagrangian function

\[
L(a_\theta, \lambda) = a^T_\theta J a_\theta - \lambda (1 - a^T_\theta a_\theta),
\]  

(4.20)

such that the direction of minimum energy, \( \bar{a}_\theta \), is given by

\[
\bar{a}_\theta = \arg\min_{a_\theta} L(a_\theta, \lambda).
\]  

(4.21)

The Lagrange parameter \( \lambda \) must be chosen so that the gradient of \( L \) with respect to \( a_\theta \) equals zero

\[
\frac{\partial L(a_\theta, \lambda)}{\partial a_i} = 2 \sum_{k=1}^{n} J_{[i,k]} a_k - 2 \lambda a_i = 0.
\]  

(4.22)
4. Structure Tensor

Combining the \( n \) equations in (4.22) yields the following equation system

\[
Ja_\theta = \lambda a_\theta.
\]

Thus, the minimization reduces to an eigenvalue problem of the symmetric matrix \( J \). Substituting (4.23) into (4.19) gives

\[
\arg\min_{\theta} E(\theta) = a^T_{\theta}Ja_\theta = a^T_{\theta}\lambda a_\theta = \lambda,
\]

and the minimum is reached when \( a_\theta \) is the eigenvector to the minimum eigenvalue of \( J \).

4.3.1. Eigenvalue analysis

The local signal model can be determined by the number of eigenvalues below a threshold, \( \gamma \), determined by the noise level in the image sequence (Jähne 2005). Without loss of generality the eigenvalues are sorted \( \lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0 \) with corresponding eigenvectors \( v_i \).

**i0D** If all eigenvalues are below \( \gamma \), the energy is constant in all direction and no optical flow can be calculated. Since the trace of a matrix is equal to the sum of its eigenvalues, this condition can be checked without doing a full eigenvalue analysis

\[
\text{tr}(J) = \sum_{i=1}^{n} J_{[i,i]} < \gamma. \tag{4.25}
\]

When (4.25) does not hold, a full eigenvalue analysis is performed.

**i1D** With only \( \lambda_1 > \gamma \), the local signal is intrinsic 1D. This is the aperture problem, and only the normal flow \( f_\perp \) can be calculated. The eigenvector \( v_1 = \begin{bmatrix} v_{1,x} & v_{1,y} & v_{1,t} \end{bmatrix}^T \) is normal on the space spanned by \( v_2 \) and \( v_3 \)

\[
f_\perp = \begin{bmatrix} v_{1,t} \
\|v_{1,x} v_{1,y}\|_2 \
\end{bmatrix} \begin{bmatrix} v_{1,x} \
\end{bmatrix} = \frac{v_{1,t}}{v_{1,x}^2 + v_{1,y}^2} \begin{bmatrix} v_{1,x} \
\end{bmatrix}. \tag{4.26}
\]

**i2D** With \( \lambda_1 \geq \lambda_2 > \gamma \), the local signal is intrinsic 2D, and the optical flow, \( f \), is uniquely determined. Flow is calculated from \( v_3 \) as in (2.28). Let \( \varphi_x \) and \( \varphi_y \) be the angle between the \( v_{3,t} \) and respectively \( v_{3,x} \) and \( v_{3,y} \). Then the calculation of \( f \) becomes

\[
f = \begin{bmatrix} -\tan \varphi_x \\
-\tan \varphi_y \\
\end{bmatrix} = \begin{bmatrix} v_{3,x} \\
v_{3,t} \\
v_{3,y} \\
\end{bmatrix} = \frac{1}{v_{3,t}} \begin{bmatrix} v_{3,x} \\
v_{3,y} \\
\end{bmatrix}. \tag{4.27}
\]
4.3. Optical flow from the 3D structure tensor

With all three eigenvalues greater than \( \gamma \), the motion is non-linear, i.e. (4.21) has no solution. No useful optical flow estimate can be calculated.

4.3.2. Confidence measure

The purpose of a confidence measure is to indicate the reliability of the computed flow. Several confidence measures based on the structure tensor has been proposed by Jähne (2005) and Barron, Fleet, and Beauchemin (1994).

Spatial gradient

The trace of the upper \( 2 \times 2 \) subtensor, corresponding to the sum of the squared spatial gradient, is a better confidence measure than the trace of the whole tensor, since the temporal gradient is not independent of velocity (Jähne 2005)

\[
\psi_{\text{grad}} = \langle g_x^2 \rangle_w + \langle g_y^2 \rangle_w. \tag{4.28}
\]

One problem with (4.28) is that it does not lie in \([0, 1]\), and therefore is unsuitable as a weight function. This can be remedied by

\[
\psi_g = 1 - \frac{1}{1 + \alpha \psi_{\text{grad}}}, \tag{4.29}
\]

which is close to zero when \( \psi_{\text{grad}} \) is low, and close to one when \( \psi_{\text{grad}} \) is high. The \( \alpha \) is a parameter connected with the total energy in the signal, such that \( \psi_g \) can reach close to one.

Size of smallest eigenvalue

The smaller \( \lambda_3 \) is, the more likely it is that the signal is locally flat in some direction, i.e. the likelihood of the signal not being \( i3D \)

\[
\psi_e = \frac{1}{(1 + \alpha \lambda_3)^2}. \tag{4.30}
\]

The \( \alpha \) is a parameter connected with the total energy in the signal, such that \( \psi_e \) can reach close to zero.

Total coherence

The total coherence measure is based on the same idea as \( \psi_e \), but does not rely on a tunable parameter

\[
\psi_t = \left( \frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3} \right)^2. \tag{4.31}
\]

It is close to one when \( \lambda_1 \gg \lambda_3 \), and zero when \( \lambda_1 \approx \lambda_3 \). Similar to \( \psi_e \) it cannot distinguish between the aperture problem, and distributed spatial structure.
4. Structure Tensor

Spatial coherence

The spatial coherence measure is used to distinguish between the aperture problem, and distributed spatial structure

\[ \psi_s = 1 - \left( \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2. \]  (4.32)

It is close to one when \( \lambda_1 \approx \lambda_2 \), i.e. no aperture problem, and zero when \( \lambda_1 \gg \lambda_2 \). In between the two extremities, the lower bound is \( 1 - \psi_t \), as the presence of an aperture problem can not be detected more reliably than the overall certainty (Jähne, Haussecker, and Geissler 1999), because \( \lambda_2 \geq \lambda_3 \). It can not distinguish between \( i^2 \text{D} \) and \( i^3 \text{D} \) signals.

Corner measure

By combining \( \psi_t \) and \( \psi_s \), and subtracting one in order to keep the result in \([0, 1]\), we get

\[ \psi_c = \psi_t + \psi_s - 1 = \left( \frac{\lambda_1 - \lambda_3}{\lambda_1 + \lambda_3} \right)^2 - \left( \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2} \right)^2 \]  (4.33)

It is close to one when the signal is \( i^2 \text{D} \), and zero otherwise.

4.4. Structure tensor in the Lucas Kanade method

By comparing (2.14) with (4.10), we get that the estimate of the Hessian in the Lucas Kanade method, is the equal to the 2\times2 upper sub-tensor of the gradient tensor. Similarly, in the case of a pure translation model, the part of (2.15) that is multiplied with \( H^{-1} \), is equal to the first two components in the third column of the gradient tensor. Thus the method by Lucas and Kanade (1981) uses five of the six elements in the gradient tensor. By assuming translation as the motion model, it is possible to reformulate (2.15) using elements from \( J \)

\[ f^{(k+1)} = f^{(k)} + \begin{bmatrix} g_x^{(k)} & g_y^{(k)} \\ g_x^{(k)} & g_y^{(k)} \end{bmatrix}_w \begin{bmatrix} g_x^{(k)} & g_y^{(k)} \\ g_y^{(k)} & g_y^{(k)} \end{bmatrix}_w^{-1} \begin{bmatrix} g_x^{(k)} & g_y^{(k)} \\ g_y^{(k)} & g_y^{(k)} \end{bmatrix}_w = f^{(k)} + \begin{pmatrix} J^{(k)}_{[2]} \\ J^{(k)}_{[3, 1:2]} \end{pmatrix}, \]  (4.34)

where the superscript indicates the iteration and the initial flow is \( f^{(0)} = 0 \). The k-th tensor is calculated from \( g \) warped with the k-th flow \( f^{(k)} \). Thus (4.34) implies that we can substitute the gradient tensor with the other structure tensor in the Lucas Kanade method.
This chapter will introduce and develop a framework for computing dense optical flow using two different methods:

- Lucas-Kanade, described in subsection 2.1.2.
- Eigenvector method, described in section 4.3.

It is used for comparing the performance of the different methods. As shown in section 4.4, Lucas-Kanade uses components from the structure tensor, and thus both methods can use the same implementation as input. Methods for calculating the structure tensor is shown in section 5.1. In section 5.2 the two methods for computing flow is described, and the overall framework is presented in section 5.3.

5.1. Structure tensor

Three methods for calculating the structure tensor is implemented:

- Gradient filters in space-time.
- Riesz transform in the Fourier domain.
- Quadrature filter in the Fourier domain.

5.1.1. Gradient tensor

The gradients is calculated using separable convolution for the each of the three direction, creating three new images $g_x, g_y, g_t$. The tensor components is then calculated with the Hadamard product, $\odot$, i.e. point-wise matrix multiplication, between the three images. Then each tensor component is smoothed with a post smooth filter $w$. The type of smoothing kernels implemented is mean and binomial (Jähne 2005). Three type of gradient kernels is implemented: Barron, Fleet, and Beauchemin (1994), Farid and Simoncelli (1997) and Scharr (2007). All the implemented filters are separable and the right side of the kernels are described in Table 5.1, with the derivate
5. Implementation

<table>
<thead>
<tr>
<th></th>
<th>$3\times3\times3$ $h_0$ and $h_1$</th>
<th>$5\times5\times5$ $h_0$ to $h_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binomial</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S$</td>
<td>$(0.5, 0.25)$</td>
<td>$(0.375, 0.25, 0.0625)$</td>
</tr>
<tr>
<td>Barron</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>$(0, 0.5)$</td>
<td>$\frac{1}{12}(0, -8, 1)$</td>
</tr>
<tr>
<td>Farid</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>$(0, 0.425287)$</td>
<td>$(0, 0.292315, 0.104550)$</td>
</tr>
<tr>
<td>$S$</td>
<td>$(0.540242, 0.229879)$</td>
<td>$(0.426375, 0.249153, 0.037596)$</td>
</tr>
<tr>
<td>Scharr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_x$</td>
<td>$(0, 0.5)$</td>
<td>$(0, 0.371, 0.0645)$</td>
</tr>
<tr>
<td>$D_t$</td>
<td>$(0, 0.5)$</td>
<td>$(0, 0.3327, 0.0836)$</td>
</tr>
<tr>
<td>$S_x$</td>
<td>$(0.6893, 0.1553)$</td>
<td>$(0.5029, 0.2331, 0.0155)$</td>
</tr>
<tr>
<td>$S_t$</td>
<td>$(0.6683, 0.1658)$</td>
<td>$(0.4702, 0.2417, 0.0232)$</td>
</tr>
</tbody>
</table>

Table 5.1.: Smoothing and derivative kernels for $3\times3\times3$ and $5\times5\times5$ neighborhoods

![Figure 5.2.: 1D illustration of determining the movement of steep edge with and without a smoothing filter](image)

Figure 5.2.: 1D illustration of determining the movement of steep edge with and without a smoothing filter

kernels $D$, being anti-symmetric, i.e. $h_{-1} = -h_1$. The two dimensions not in the direction of the derivative is smoothed with the corresponding smoothing kernel $S$. The Barron derivative uses the binomial for smoothing. Note that Scharr uses different kernels for spatial and temporal filtering. See algorithm 5.1 for a pseudo-code description.

5.1.2. Riesz tensor

The calculation of the Riesz tensor is similar to the gradient tensor calculation, but with the derivative images swapped for the Riesz transformed images. Before filtering, the image is smoothed in order to better detect movement of steep edges, see Figure 5.2 for an illustration. The Riesz transform is done in the Fourier domain with the filter kernel as in (3.19). See algorithm 5.2 for a pseudo-code description.
5.1. Structure tensor

**Algorithm 5.1**: Gradient tensor

**Input**: Space-time image \( g \), Gradient kernels \( d_x, d_y, d_t \), Smoothing kernel \( w \)

**Output**: List of structure tensors \( J \)

**begin**

// Filter image
for all \( x \in \mathbb{R} \):
\[
\begin{bmatrix}
g_x(x) & g_y(x) & g_t(x)
g_x(x) & g_y(x) & g_t(x)
g_x(x) & g_y(x) & g_t(x)
\end{bmatrix} \leftarrow \begin{bmatrix}
(g \ast d_x)(x) & (g \ast d_y)(x) & (g \ast d_t)(x)
(g \ast d_x)(x) & (g \ast d_y)(x) & (g \ast d_t)(x)
(g \ast d_x)(x) & (g \ast d_y)(x) & (g \ast d_t)(x)
\end{bmatrix}
\]

// Calculate the six component of the structure tensor
\[
J \leftarrow \begin{bmatrix}
g_x \ast g_x & g_x \ast g_y & g_x \ast g_t 
g_y \ast g_x & g_y \ast g_y & g_y \ast g_t 
g_t \ast g_x & g_t \ast g_y & g_t \ast g_t
\end{bmatrix}
\]

// Smooth the structure tensor
for all \( J_{ij}, j > i \):
\[
J_{ij} \leftarrow J_{ij} \ast w
\]

**Algorithm 5.2**: Riesz tensor

**Input**: Space-time image \( g \), Pre smoothing kernel \( h \), Post smoothing kernel \( w \)

**Output**: List of structure tensor \( J \)

**begin**

// Calculate filters
\[
k \leftarrow \begin{bmatrix} k_x & k_y & \omega \end{bmatrix}^T, \ k_x, k_y, \omega \in [-\pi : \pi]
\]

\[
\begin{bmatrix} \mathcal{R}_x & \mathcal{R}_y & \mathcal{R}_t \end{bmatrix} \leftarrow \text{ifftshift} \left( \begin{bmatrix} -j k_x / ||k|| & -j k_y / ||k|| & -j k_t / ||k|| \end{bmatrix} \right)
\]

// Filter image in Fourier domain
\[
g \leftarrow g \ast h
\]

\[
\hat{g} \leftarrow \text{fft}(g)
\]

\[
\begin{bmatrix} g_x & g_y & g_t \end{bmatrix} \leftarrow \text{real} \left( \text{ifft} \left( \begin{bmatrix} \mathcal{R}_x \ast \hat{g} & \mathcal{R}_y \ast \hat{g} & \mathcal{R}_t \ast \hat{g} \end{bmatrix} \right) \right)
\]

// Calculate the six component of the structure tensor
\[
J \leftarrow \begin{bmatrix}
g_x \ast g_x & g_x \ast g_y & g_x \ast g_t 
g_y \ast g_x & g_y \ast g_y & g_y \ast g_t 
g_t \ast g_x & g_t \ast g_y & g_t \ast g_t
\end{bmatrix}
\]

// Smooth the structure tensor
for all \( J_{ij}, j > i \):
\[
J_{ij} \leftarrow J_{ij} \ast w
\]
5. Implementation

5.1.3. Quadrature tensor
The Quadrature tensor is calculated by filtering in the Fourier domain as described in subsection 4.2.3. The selection of filter kernels for pre-smoothing is the same as for post-smoothing. The image is smoothed before filtering for the same reason as in the Riesz tensor. See algorithm 5.3 for a pseudo-code description.

5.2. Flow calculation
Two algorithms for calculating flow is implemented: Lucas-Kanade described in subsection 2.1.2 and the eigenvector method described in section 4.3. Both methods uses eigenvectors to identify the presence of an aperture problem.

5.2.1. Lucas Kanade
The Lucas-Kanade method is implemented using matrix inversion for solving (4.34) for each pixel in an image region $R$. It is only performed if the upper $2 \times 2$ sub-tensor $J_2$ is invertible. The presence of an aperture problem is determined by the number of eigenvalues from $J_2$ that is above a threshold $\tau$. If only one eigenvalue is larger than $\tau$, the calculated flow is projected into the direction of the eigenvector corresponding to the largest eigenvalue. See algorithm 5.4 for a pseudo-code description.

5.2.2. Eigenvector method
The eigenvector method differs from the Lucas Kanade method by using the eigenvalues from the complete structure tensor for estimating the intrinsic dimensionality of the signal. Therefore setting the threshold $\tau$ also implies an upper bound on when the flow is computed. if 1 or 2 eigenvalues are larger than $\tau$, it uses (4.26) and (4.27) to calculate the flow, respectively. See algorithm 5.5 for a pseudo-code description.

5.3. Framework
Several of the techniques from section 2.1 has been used to increase the accuracy of the calculated flow:

- Coarse-to-fine Laplacian pyramid.
- Spline interpolation of the warped image.
- Iterative update of the flow estimate.
Algorithm 5.3: Quadrature tensor

**Input:** Space-time image $g$, Pre smoothing kernel $h$, Post smoothing kernel $w$, Center frequency $k_0$, Bandwidth $B$

**Output:** List of structure tensor $J$

**begin**

// Initialization
\[ \alpha \leftarrow \frac{3}{4}, \beta \leftarrow \frac{1}{4}, a \leftarrow 2, b \leftarrow 1 + \sqrt{5}, c \leftarrow \sqrt{10 + 2\sqrt{5}} \]
\[ \begin{bmatrix} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \end{bmatrix} \leftarrow c \begin{bmatrix} a & -a & b & b & 0 & 0 \\ 0 & 0 & a & -a & b & b \\ b & b & 0 & 0 & a & -a \end{bmatrix} \] // (4.14)

// Calculate filters
\[ k \leftarrow [k_x \ k_y \ \omega]^{T}, k_x, k_y, \omega \in [-1 : 1] \]
\[ \tilde{k} \leftarrow \frac{k}{||k||} \]
\[ R \leftarrow \exp \left( \frac{\log(||k||) - \log(k_0)}{(B/2)^2 \log 2} \right) \] // (4.12)

for $i \in \{1, \ldots, 6\}$:
\[ D_i \leftarrow \left( \tilde{k} d_i \right)^2 \left\lfloor \text{sgn}(k^T d_i) \right\rfloor \] // (4.13)
\[ \hat{Q}_i \leftarrow \text{ifftshift}(R \odot D_i) \] // (4.11)
\[ M_i \leftarrow \alpha d_i d_i^T - \beta I \]

// Filter image in Fourier domain
\[ g \leftarrow g \ast h \] // Smooth the image before filtering
\[ \hat{g} \leftarrow \text{fft}(g) \]
forall $Q_i$:
\[ q_i \leftarrow \text{abs}(\text{ifft}(\hat{g} \odot \hat{Q}_i)) \] // (4.16)

forall $x$:
for $i \in \{1, \ldots, 6\}$:
\[ J[x] \leftarrow J[x] + q_i[x] M_i \] // (4.17)

// Smooth the structure tensor
forall $J_{ij}$, $j > i$:
\[ J_{ij} \leftarrow J_{ij} \ast w \]
5. Implementation

Algorithm 5.4: Lucas Kanade

**Input:** List of structure tensors \( J \) for each pixel in Image region \( R \), Threshold \( \tau \)

**Output:** Flow \( f \)

**begin**

forall \( x \in R \):

\[
A \leftarrow \begin{bmatrix} J_{xx}[x] & J_{xy}[x] \\ J_{xy}[x] & J_{yy}[x] \end{bmatrix}
\]

\[
b \leftarrow \begin{bmatrix} J_{xx}[x] \\ J_{yy}[x] \end{bmatrix}
\]

\( \lambda_1, \lambda_2, \nu_1, \nu_2 \leftarrow \text{eig}(A) \)

if \( \det A \neq 0 \):

if \( \lambda_1 > \tau \):

if \( \lambda_2 > \tau \):

// i2D Structure, no change in \( f \)

else:

// i1D structure

\[
f[x] \leftarrow \langle f[x], \nu_1 \rangle \nu_1
\]

else:

// i0D structure

\[
f[x] \leftarrow 0
\]

else:

\[
f[x] \leftarrow 0
\]

**end**

Algorithm 5.5: Flow from eigenvectors

**Input:** List of structure tensors \( J \), Image region \( R \), Threshold \( \tau \)

**Output:** Flow \( f \)

**begin**

forall \( x \in R \):

\( \lambda_1, \lambda_2, \lambda_3, \nu_1, \nu_2, \nu_3 \leftarrow \text{eig}(J[x]) \)

if \( \lambda_1 > \tau \):

if \( \lambda_2 > \tau \):

if \( \lambda_3 > \tau \):

// i3D structure

\[
f[x] \leftarrow 0
\]

else:

// i2D structure

\[
f[x] \leftarrow \frac{1}{\nu_{l,t}} \begin{bmatrix} \nu_{3,x} \\ \nu_{3,y} \end{bmatrix}
\]

else:

// i1D structure

\[
f[x] \leftarrow \frac{\nu_{1,t}}{\nu_{1,x} + \nu_{1,y}} \begin{bmatrix} \nu_{1,x} \\ \nu_{1,t} \end{bmatrix}
\]

else:

// i0D structure

\[
f[x] \leftarrow 0
\]

**end**
Algorithm 5.6: Flow framework

Input: Grayscale space-time image $g$
Output: Flow $f$

begin
    $g^{(n)} \leftarrow \text{pyramid}(g, \text{top layer})$
    for $n \in \{\text{top layer}, \ldots, 0\}$ :
        $f^{(0)}_n \leftarrow 2 \times \text{pyramid}_\text{expand}(f^{(k)}_{n+1})$  // flow from lower layer, initialized to 0
    while $k < \text{max iterations}$ :
        repeat
            $g_w \leftarrow \text{warp}(g^{(n)}, f^{(k-1)}_n)$
            $g_u \leftarrow \text{upsample}_\text{in_time}(g_w)$
            $J \leftarrow \text{tensor}(g_u)$  // See section 5.1
            $\Delta f^{(k)}_n \leftarrow \text{flow}_\text{calc}(J, \mathcal{R}^{(k-1)})$  // See section 5.2
            $f^{(k)}_n \leftarrow f^{(k-1)}_n + \Delta f^{(k)}_n$
            $\mathcal{R}^{(k)} \leftarrow \{x \in \mathcal{R}^{(k-1)} \mid \Delta f^{(k)}[x] < \varepsilon\}$
        until $\mathcal{R}^{(k)} \subset \emptyset$
end

The pyramid implementation is from scikit-image (Walt et al. 2014). Warp is done using RectBivariateSpline() from SciPy (Jones, Oliphant, Peterson, et al. 2001–). In order to create enough support in time for the Fourier-based filters to work, the warped image is upsampled. The upsample rate is a tunable parameters. See algorithm 5.6 for a pseudo-code description.
6 Experiments

In order to measure the performance of the different optical flow method, experiments are set up on three datasets:

- A synthetic sequence without noise.
- The Middlebury dataset containing both synthetic sequences and real world images.
- The Leuven dataset containing realistic synthetic echocardiograms.

6.1. Error metrics

The most commonly used error metric for measuring performance of optical flow algorithms is the angular error (AE) (Baker, Scharstein, et al. 2011). The AE between a flow vector $f = (u, v)$ and the ground-truth $f_{gt} = (u_{gt}, v_{gt})$ is the angle in 3D space between $(u, v, 1)$ and $(u_{gt}, v_{gt}, 1)$. The AE is computed by taking the inner product between the two extended vectors, dividing by the product of their norms, and then taking the arcus cosine:

$$AE = \arccos \left( \frac{\langle f, f_{gt} \rangle + 1}{\sqrt{\|f\|^2 + 1} \sqrt{\|f_{gt}\|^2 + 1}} \right) = \arccos \left( \frac{u_{gt}u + v_{gt}v + 1}{\sqrt{u^2 + v^2 + 1} \sqrt{u_{gt}^2 + v_{gt}^2 + 1}} \right)$$  (6.1)

The AE was first used by Fleet and Jepson (1990), and made popular by the seminal work of Barron, Fleet, and Beauchemin (1994). It is a relative measure, with an arbitrary scaling constant of 1 to convert the units from pixels to degrees, and to avoid dividing by zero. One downside with the AE is that it penalizes large flows less than small. Given a ground-truth $f_{gt} = (1.5, 0)$ and two flows $f_1 = (1, 0)$ and $f_2 = (2, 0)$, the corresponding AE are $AE_1 = 11.3^\circ$ and $AE_2 = 7.1^\circ$ respectively.

An alternative error measure is the L2-distance between the ground-truth and estimated flow, introduced by Otte and Nagel (1994) and called the endpoint error (EE):

$$EE = \|f - f_{gt}\|_2 = \sqrt{(u - u_{gt})^2 + (v - v_{gt})^2}$$  (6.2)
6. Experiments

<table>
<thead>
<tr>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>threshold $\tau$</td>
<td>9e-6</td>
</tr>
<tr>
<td>Up sample rate</td>
<td>4</td>
</tr>
<tr>
<td>Layers</td>
<td>2</td>
</tr>
<tr>
<td>Post-Smooth $w$ Mean</td>
<td>7x7x3</td>
</tr>
<tr>
<td>Pre-smooth $h$ None</td>
<td>None</td>
</tr>
<tr>
<td>Gradient kernel Scharr</td>
<td>5x5x5</td>
</tr>
<tr>
<td>Bandwidth $B$ 1.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Center freq $k_0$</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Table 6.1.: Parameters for the rectangular box experiments

The EE is an absolute measure, and hence does not favor one type of error over the other. Thus the endpoint error for $f_1$ and $f_2$ is both 0.5 pixels.

6.1. Statistics

For estimated flow fields, we calculate the mean, the standard deviation and robustness statistics in line with the work of Baker, Scharstein, et al. (2011). In particular $R_X$ is the percentage of pixels with error measure above X. For the angular error (AE), $R_{10.0}$, $R_{5.0}$, and $R_{2.5}$ are computed; for the endpoint error (EE), $R_{2.0}$, $R_{1.0}$ and $R_{0.5}$ are computed.

6.2. Rectangular boxes

In order to verify that the quadrature and Riesz based methods perform comparable to the gradient based method, a synthetic test sequence containing two rectangular boxes were created. The right box, moving with $(2, 2)$, is occluded by the left box, moving with $(-1, -1)$ see Figure 6.2d-f for the three frames used in the experiment. The size of the frames are 192x128 and the grayscale values are floating points from 0 to 1. The parameters in Table 6.1 was tuned to obtain minimum endpoint error. See section A.1 for details of the parameter search. The left box moves with $(2, 2)$ and the right box moves with $(-1, -1)$, the ground truth is visualised in Figure 6.2b, and only pixels inside the mask in Figure 6.2c are used for calculating statistics.
6.3. Middlebury

Baker, Scharstein, et al. (2011) published a database\(^1\) of 8 sequences with ground truth data available, often referred to as the Middlebury dataset. Five of the sequences are synthetic, and the other three are real world images with ground truth generated from hidden fluorescent texture painted on the scene. It has been highly influential on the evaluation of optical flow methods. Figure 6.3 shows example frames from three of the sequences. However, only one frame for each of the sequences has ground truth available. Input to the framework consists of three frames, and flow is estimated for the center frame. Parameters was tuned for minimum EE on the Urban2 sequence, see section A.2 for details.

6.4. Leuven

Alessandrini, Chakraborty, et al. (2018) published a database\(^2\) containing realistic synthetic echocardiograms with ground truth data. The availability of high quality synthetic echocardiograms makes it easier to validate performance, as there is no need to design a complex simulation setup, and it gives the possibility to compare results. It contains 3 apical views of

\(^1\)http://vision.middlebury.edu/flow/
\(^2\)https://gbiomed.kuleuven.be/english/research/50000635/50508167/open-data
6. Experiments

![Example frames from Middlebury dataset](image)

(a) Urban2  (b) Grove2  (c) Dimetrodon

Figure 6.3.: Example frames from Middlebury dataset

<table>
<thead>
<tr>
<th></th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threshold $r$</td>
<td>1e-6</td>
<td>7e-6</td>
</tr>
<tr>
<td>Up sample rate</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Layers</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Post-Smooth $w$</td>
<td>Mean $7\times7\times3$</td>
<td>Mean $7\times7\times3$</td>
</tr>
<tr>
<td>Pre-smooth $h$</td>
<td>Mean $7\times7\times3$</td>
<td>Mean $7\times7\times3$</td>
</tr>
<tr>
<td>Gradient kernel</td>
<td>Binom $3\times3\times3$</td>
<td>Binom $3\times3\times3$</td>
</tr>
<tr>
<td>Bandwidth $B$</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Center freq $k_0$</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Table 6.4.: Parameters for Middlebury experiments
Figure 6.5.: Frame 8 of apical 2 chamber view with normal motion on GE Vingmed Ultrasound. Ground truth arrows are scaled with a factor 10.

the heart with 5 motion patterns for 7 vendor specific image formations, which gives a total of 105 sequences. The three apical views corresponds to the most common probe orientation. Frame 8 of the apical 2 chamber view with normal motion pattern for GE Vingmed Ultrasound, with the ground truth overlaid, is shown in Figure 6.5. Input to the framework consists of three frames, and flow is estimated for the center frame. The parameters for the experiment was tuned for minimum EE on frame 8 of apical 2 chamber view with normal motion for GE Vingmed Ultrasound, see section A.3 for details. Parameters obtained are presented in Table 6.6. A gradient setup using Barron filter kernels without upsampling is used as a baseline reference. The lack of pre-smoothing can be contributed to the fact that the images has been smoothed in the image formation pipeline. The large window size for post-smooth is caused by the sparseness of the ground truth data, as there are only 180 ground truth points per frame.
6. Experiments

Table 6.6.: Parameters for Leuven experiments

<table>
<thead>
<tr>
<th></th>
<th>Gradient</th>
<th>Scharr</th>
<th>Quadrature</th>
<th>Riesz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lucas-Kanade</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>threshold $r$</td>
<td>1e-5</td>
<td>1e-5</td>
<td>1e-3</td>
<td>1e-5</td>
</tr>
<tr>
<td>Up sample rate</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Layers</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Post-Smooth $w$</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td>15x15x7</td>
<td>15x15x7</td>
<td>15x15x7</td>
<td>15x15x7</td>
</tr>
<tr>
<td>Pre-smooth $h$</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient kernel</td>
<td>Barron</td>
<td>Scharr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3x3x3</td>
<td>3x3x3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth $B$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center freq $k_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Eigenvector</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>threshold $r$</td>
<td>5e-5</td>
<td>1e-5</td>
<td>1e-3</td>
<td>1e-5</td>
</tr>
<tr>
<td>Up sample rate</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Layers</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Post-Smooth $w$</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td>15x15x7</td>
<td>15x15x7</td>
<td>15x15x7</td>
<td>15x15x7</td>
</tr>
<tr>
<td>Pre-smooth $h$</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient kernel</td>
<td>Barron</td>
<td>Scharr</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3x3x3</td>
<td>3x3x3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth $B$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center freq $k_0$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 7

Results

This chapter presents the results obtained from the experiments described in the previous chapter.

7.1. Rectangular boxes

The estimated flow for the six methods is visualized in Figure 7.1a-f, color coded according to the color wheel in Figure 6.2a. The statistics is reported in Table 7.2. The corners of the boxes is distinct from the sides due to the presence of the aperture problem. The biggest difference visually is in the area of low contrast at the bottom of the right box, where the gradient based methods has worse results than the four other methods. The method with lowest mean AE and EE is Lucas-Kanade with Riesz tensor. The standard deviation is smallest for Lucas-Kanade with quadrature tensor for both AE and EE, but not by much. The eigenvector methods show higher values for all statistics compared to the Lucas-Kanade method with the same tensor. The difference in the EE robust statistic $R_X$ implies that the eigenvector method with gradient and Riesz tensor have a heavier tail to the right than the Lucas-Kanade method.

The overall difference in the statistics between the methods is quite small. In conclusion, the quadrature and Riesz based methods perform on a par with, and in some areas even better than, the gradient methods.

7.2. Middlebury

The AE and EE for each sequence is shown in Figure 7.3. Mean of the statistics over the 8 sequences for each method is shown in Table 7.4, see section B.1 for the full statistics for each sequence. The Lucas-Kanade methods outperforms the eigenvector methods on both AE and EE for every sequence. This is most likely due to the higher threshold $r$ parameter as the eigenvector methods seems to be more sensitive too it. Also the eigenvector methods has problems with sometimes creating large outliers, skewing the normal statistics such as mean and standard deviation. The large standard deviation on the endpoint errors is caused by large outliers on the Urban3 sequence. Among the Lucas-Kanade methods, the Riesz and
7. Results

Figure 7.1.: Visualization of the flow in rectangular boxes. First row is Lucas-Kanade, second row is eigenvector method. From left to right: Gradient tensor, Quadrature tensor and Riesz tensor.

<table>
<thead>
<tr>
<th></th>
<th>Lucas-Kanade</th>
<th></th>
<th>Eigenvector</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
<td>Riesz</td>
<td>Gradient</td>
</tr>
<tr>
<td>Angular error</td>
<td>32.34</td>
<td>30.87</td>
<td><strong>28.39</strong></td>
<td>33.91</td>
</tr>
<tr>
<td></td>
<td>25.56</td>
<td><strong>24.23</strong></td>
<td>26.28</td>
<td>25.82</td>
</tr>
<tr>
<td>R2.5</td>
<td><strong>85.76</strong></td>
<td>92.48</td>
<td>88.50</td>
<td>85.89</td>
</tr>
<tr>
<td>R5.0</td>
<td><strong>77.26</strong></td>
<td>85.77</td>
<td>80.63</td>
<td>77.19</td>
</tr>
<tr>
<td>R10.0</td>
<td>71.23</td>
<td>73.80</td>
<td><strong>68.16</strong></td>
<td>71.80</td>
</tr>
<tr>
<td>Endpoint error</td>
<td>1.18</td>
<td>1.17</td>
<td><strong>1.06</strong></td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>0.85</td>
<td><strong>0.78</strong></td>
<td>0.83</td>
<td>0.87</td>
</tr>
<tr>
<td>R0.5</td>
<td>71.38</td>
<td>76.23</td>
<td><strong>70.20</strong></td>
<td>71.96</td>
</tr>
<tr>
<td>R1.0</td>
<td>54.63</td>
<td>51.91</td>
<td><strong>44.84</strong></td>
<td>60.82</td>
</tr>
<tr>
<td>R2.0</td>
<td>16.65</td>
<td>15.80</td>
<td><strong>14.03</strong></td>
<td>25.42</td>
</tr>
</tbody>
</table>

Table 7.2.: Statistics from rectangular boxes experiment.
gradient based method performs equally, but are outperformed by the quadrature based method. However the standard deviation for both AE and EE is quite high. The robust statistics indicates that among the Lucas-Kanade methods, gradient and Riesz has equally shaped distributions, and quadrature has a lighter tail.

The sequences with the largest difference in AE between the methods is the Hydrangea sequence, it is visualised in Figure 7.5. All the methods performs equally well on the foreground object, but the difference is on the low texture background. None of the eigenvector methods estimated any flow on the background. Among the Lucas-Kanade methods, the quadrature method fills in almost all of the background flow.

### 7.3. Leuven

Since the framework uses three frames to calculate the motion on the center frame, this excludes frame 0 and 54. Thus the flow is estimated from frame 1 to frame 53. The sequences used are the
7. Results

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>22.14</td>
<td>18.50</td>
</tr>
<tr>
<td>std</td>
<td>29.50</td>
<td>27.40</td>
</tr>
<tr>
<td>R2.5</td>
<td>67.50</td>
<td>64.30</td>
</tr>
<tr>
<td>R5.0</td>
<td>54.51</td>
<td>50.09</td>
</tr>
<tr>
<td>R10.0</td>
<td>42.50</td>
<td>37.46</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>2.03</td>
<td>1.87</td>
</tr>
<tr>
<td>std</td>
<td>2.47</td>
<td>2.74</td>
</tr>
<tr>
<td>R0.5</td>
<td>54.56</td>
<td>48.62</td>
</tr>
<tr>
<td>R1.0</td>
<td>40.63</td>
<td>34.56</td>
</tr>
<tr>
<td>R2.0</td>
<td>28.44</td>
<td>22.66</td>
</tr>
</tbody>
</table>

Table 7.4.: Statistics from the Middlebury experiment. The number for each measure is the mean over the 8 sequences in the Middlebury dataset.

apical 2, 3, and 4 chamber views, with normal motion on GE Vingmed Ultrasound. The motion is largest at frame 3 to 10 and smallest at frame 40 to 46. At frame 20 to 30 there is a motion at approximately half the size of the largest motion. See Figure 7.6 for a visualization of the mean magnitude of the ground truth for A2C. The mean AE and EE is shown in Figure 7.7, Figure 7.8 and Figure 7.9, see section B.2 for the rest of the statistics.

During the larger motions, the baseline gradient methods are inferior. In the frames with smaller motion they sometimes the superior methods, e.g. frame 40 in Figure 7.9, but the difference is small at around 0.1 pixels. Except for the baseline gradient, the difference between Lucas-Kanade and eigenvector methods with the same structure tensor is negligible. Overall the best performing methods are the Riesz and Scharr based methods. The quadrature tensor has higher mean AE and EE on the parts with large motion, however the difference is quite small, and a lot smaller than the standard deviation at the same frame.
Figure 7.5.: Estimated flow for the Hydrangea sequence from the Middlebury dataset.

Figure 7.6.: Mean magnitude of the ground truth for apical 2 chamber view on GE Vingmed Ultrasound
7. Results

Figure 7.7.: Mean errors for apical 2 chamber view on GE Vingmed Ultrasound
Figure 7.8.: Mean errors for apical 3 chamber view on GE Vingmed Ultrasound
7. Results

Figure 7.9.: Mean errors for apical 4 chamber view on GE Vingmed Ultrasound
8 Discussion

8.1. Homogeneous areas

As seen in the results, both the Riesz transform and quadrature filters are viable alternatives to gradient filters for calculating the structure tensor. The largest difference in performance was in the homogeneous areas on the images from the Middlebury dataset, where the Lucas-Kanade with quadrature tensor outperformed the other methods. Due to the speckle noise, there are few homogeneous regions in echocardiograms. This leads to the disappearance of the quadratures tensors advantage, and it even falls a bit behind the Riesz tensor and the gradient tensor calculated with the Scharr operator. The Riesz tensor handled the low contrast region on the rectangular boxes sequence better than the gradient, and performed equally well on the echocardiograms.

8.2. Parameter tuning

A time consuming part of the experiment setup has been the parameter tuning, and thus a complete search of the parameter space is infeasible. For the Leuven experiment alone, a total of over 6000 different parameter combinations was tested.

8.3. Parameter sensitivity

The eigenvector method is more sensitive to the threshold parameter, and due to this it captures less movements. When the threshold is too low, it sometimes produces large outliers. The lack of a confidence measures in the framework is also a possible contributing factor to the sensitivity.

8.4. Computational cost

The quadrature tensor has the highest numbers of parameters, and also the highest computational cost due to the use of six filters, while the other two tensor types use three. The lowest
computational cost is the gradient tensor, since it uses separable convolution.

### 8.5. Upsampling of the image sequence

Initially the upsampling in time was implemented in order to give the Fourier based method enough support in time, but during parameter tuning it showed increased performance on the gradient methods also. As the Lucas-Kanade method works best with small movements, and upsampling in time gives lower inter-frame movement, this leads to better results during large motions. In frames with small motion the methods with upsampled input was outperformed by the only non-upsampling estimation, the baseline Barron filters. The largest improvement from upsampling is in the eigenvector method, where a larger range of the threshold parameter yields low EE. Thus upsampling reduces the value of the smallest eigenvalue and affects the largest eigenvalue to a lesser extent.

### 8.6. Drawbacks with the framework

During the experiments, the flow framework described in section 5.3 was shown to have a couple of drawbacks.

#### 8.6.1. Global warp

The biggest one is the use of a global warp, which leads to a problem with convergence on boundaries in the flow field. This is visualised in Figure 8.1, where a square box is moving with ground truth $(1, 1)$. The ground truth magnitude is 1.41, but in a ring around the corners some of the estimation reaches a magnitude of over 3 after 10 iterations. Thus all the experiments is run with only one iteration at each scale. One alternative is to warp the whole image for each of the flow points, but this has high computational cost, as it would lead to $O(n^2)$ number of FFTs to computed for a $n \times n$ sized region.

#### 8.6.2. No confidence measures

The flow framework does not use any of the confidence measures described in subsection 4.3.2, leading to possible unreliable flows.
8.6. Drawbacks with the framework

(a) Magnitude of initial flow estimation

(b) Magnitude after 10 iterations

Figure 8.1.: Visualization of the problems caused by a global warp
9 Concluding remarks

9.1. Summary

Two methods for calculating optical flow from grayscale images has been presented: Lucas-Kanade and the eigenvector method. Both uses components from the structure tensor for the computation. Three methods for calculating the structure tensor was shown: gradient filters, quadrature filters, and the Riesz transform. In order to evaluate the performance of the different methods with the different structure tensors, a framework was implemented. Experiments was done on three datasets: a synthetic no-noise image containing two rectangular boxes moving in opposite direction with different velocities; the Middlebury dataset containing a total for 8 sequences; Realistic synthetic echocardiograms with sequences representing three different acquisition orientations. The Lucas-Kanade method was easier to tune, and performed overall better than the eigenvector method. On the rectangular boxes the best performing method was the Riesz tensor. On the Middlebury dataset the quadrature tensor gave best performance due to its good performance in homogeneous regions. Both the Riesz tensor and the quadrature tensor performed equal to the gradient tensor on echocardiograms, but have higher computational cost. Thus no significant evidence in support of replacing the gradient with the other methods for doing speckle tracking in echocardiography has been found.

9.2. Future work

9.2.1. Region based error

Calculate the error for each region of the heart instead of over the heart as a whole, as different parts of the heart has different movement patterns, possibly leading to different methods being preferred.

9.2.2. Adaptive parameter selection

Currently the threshold parameter is a hand tuned parameter. This is a time consuming process, and only one set of parameters is used for the entire image. A method for adaptively select
9. Concluding remarks

the parameters should give increased performance, e.g. it could increase the window size in homogeneous regions or select the threshold based on the local signal energy.

9.2.3. Space-time implementation of Riesz transform

The Riesz transform in subsection 5.1.2 is implemented in the Fourier domain. A separable space-time implementation of (3.22) could possibly be used to reduce the computational cost of the Riesz tensor.

9.2.4. Local warp

As described in subsection 8.6.1 the global warp prevents the iterative update to work. In implementation with local warp, this problem would not exist.

9.2.5. Complex motion models

Since the actual motion of the heart muscle is not just translational, the use of a more complex motion model could possible give higher performance.

9.2.6. Use of different pre-smoothing filters for the Riesz transform

The gradient tensor computed with the Scharr filters outperformed the one based on Barron filters. In $3\times3\times3$ neighborhoods, the only difference between them is the smoothing filter. This leads us to believe that smoothing filters specially designed for the Riesz transform has potential to increase its performance.
In this chapter the tuning of the parameters for the experiments in chapter 6 is described. A search over the threshold parameter with different pre-smoothing and gradient filters are done. For the quadrature filter only the threshold search with the same center frequency $k_0$ and up sample rate as the one selected for the experiment is shown.

A.1. Square boxes

Figure A.1.: Tuning the Lucas-Kanade method with gradient tensor
A. Parameter tuning

Figure A.2.: Tuning the Lucas-Kanade method with Riesz tensor

Figure A.3.: Tuning the Lucas-Kanade method with quadrature tensor with $k_0 = 0.7$ and up sample rate of 8
A.1. Square boxes

Figure A.4.: Tuning the eigenvector method with gradient tensor

Figure A.5.: Tuning the eigenvector method with Riesz tensor
A. Parameter tuning

Figure A.6.: Tuning the eigenvector method with quadrature tensor with $k_0 = 0.7$ and upsample rate of 8

A.2. Middlebury

Figure A.7.: Tuning the Lucas-Kanade method with gradient tensor

Figure A.8.: Tuning the Lucas-Kanade method with Riesz tensor
A.2. Middlebury

Figure A.9.: Tuning the Lucas-Kanade method with quadrature tensor with $k_0 = 0.5$ and up sample rate of 8

Figure A.10.: Tuning the eigenvector method with gradient tensor

Figure A.11.: Tuning the eigenvector method with Riesz tensor
A. Parameter tuning

Figure A.12.: Tuning the eigenvector method with quadrature tensor with $k_0 = 0.5$ and up sample rate of 8

A.3. Leuven

Figure A.13.: Tuning the Lucas-Kanade method with gradient tensor
Figure A.14.: Tuning the Lucas-Kanade method with Riesz tensor

Figure A.15.: Tuning the Lucas-Kanade method with quadrature tensor with $k_0 = 0.7$ and upsample rate of 8
Figure A.16.: Tuning the eigenvector method with gradient tensor

Figure A.17.: Tuning the eigenvector method with Riesz tensor
Figure A.18: Tuning the eigenvector method with quadrature tensor with $k_0 = 0.7$ and up sample rate of 8
B Statistics

B.1. Middlebury

In this section the complete statistics from the Middlebury experiment is shown. A plot comparing the standard deviation of both AE and EE is shown before the tables with the complete statistics.

![Angular error standard deviation](image)

Figure B.1.: Angular error standard deviation
B. Statistics

Figure B.2.: Endpoint error standard deviation

Table B.3.: Dimetrodon

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td>22.39</td>
<td>18.82</td>
</tr>
<tr>
<td><strong>std</strong></td>
<td>23.55</td>
<td>24.22</td>
</tr>
<tr>
<td>R2.5</td>
<td>68.07</td>
<td>70.04</td>
</tr>
<tr>
<td>R10</td>
<td>50.58</td>
<td>43.07</td>
</tr>
<tr>
<td>R5</td>
<td>58.55</td>
<td>55.03</td>
</tr>
<tr>
<td><strong>Endpoint error</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td>0.80</td>
<td>0.78</td>
</tr>
<tr>
<td><strong>std</strong></td>
<td>0.79</td>
<td>1.21</td>
</tr>
<tr>
<td>R0.5</td>
<td>51.95</td>
<td>42.99</td>
</tr>
<tr>
<td>R1</td>
<td>35.35</td>
<td>30.16</td>
</tr>
<tr>
<td>R2</td>
<td>11.61</td>
<td>9.68</td>
</tr>
</tbody>
</table>
### B.1. Middlebury

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>22.40</td>
<td>12.36</td>
</tr>
<tr>
<td>std</td>
<td>31.01</td>
<td>22.68</td>
</tr>
<tr>
<td>R2.5</td>
<td>53.48</td>
<td>44.25</td>
</tr>
<tr>
<td>R5</td>
<td>43.90</td>
<td>32.12</td>
</tr>
<tr>
<td>R10</td>
<td>37.39</td>
<td>24.17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>1.09</td>
<td>0.69</td>
</tr>
<tr>
<td>std</td>
<td>1.32</td>
<td>1.05</td>
</tr>
<tr>
<td>R0.5</td>
<td>42.41</td>
<td>29.82</td>
</tr>
<tr>
<td>R1</td>
<td>35.10</td>
<td>21.58</td>
</tr>
<tr>
<td>R2</td>
<td>28.36</td>
<td>14.51</td>
</tr>
</tbody>
</table>

Table B.4.: Grove2

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>16.57</td>
<td>14.77</td>
</tr>
<tr>
<td>std</td>
<td>26.01</td>
<td>24.44</td>
</tr>
<tr>
<td>R2.5</td>
<td>65.19</td>
<td>62.36</td>
</tr>
<tr>
<td>R5</td>
<td>49.75</td>
<td>45.90</td>
</tr>
<tr>
<td>R10</td>
<td>33.97</td>
<td>30.34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>1.64</td>
<td>1.46</td>
</tr>
<tr>
<td>std</td>
<td>2.26</td>
<td>2.15</td>
</tr>
<tr>
<td>R0.5</td>
<td>51.58</td>
<td>47.53</td>
</tr>
<tr>
<td>R1</td>
<td>38.82</td>
<td>34.81</td>
</tr>
<tr>
<td>R2</td>
<td>27.52</td>
<td>23.94</td>
</tr>
</tbody>
</table>

Table B.5.: Grove3
## B. Statistics

<table>
<thead>
<tr>
<th></th>
<th>Angular error</th>
<th>Endpoint error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lucas-Kanade</td>
<td>Eigenvector</td>
</tr>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>27.11</td>
<td>11.28</td>
</tr>
<tr>
<td>std</td>
<td>31.10</td>
<td>20.65</td>
</tr>
<tr>
<td>R10</td>
<td>49.68</td>
<td>26.26</td>
</tr>
<tr>
<td>R2.5</td>
<td>74.37</td>
<td>47.22</td>
</tr>
<tr>
<td>R5</td>
<td>63.41</td>
<td>37.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>std</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0.5</td>
<td>71.20</td>
<td>22.31</td>
<td>1.79</td>
<td>1.56</td>
</tr>
<tr>
<td>R1</td>
<td>53.41</td>
<td>22.78</td>
<td>1.19</td>
<td>1.55</td>
</tr>
<tr>
<td>R2</td>
<td>38.70</td>
<td>43.90</td>
<td>1.75</td>
<td>1.55</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>20.28</th>
<th>22.78</th>
<th>57.38</th>
<th>59.16</th>
</tr>
</thead>
<tbody>
<tr>
<td>R5</td>
<td>45.30</td>
<td>42.80</td>
<td>64.00</td>
<td>75.02</td>
</tr>
<tr>
<td>R10</td>
<td>26.67</td>
<td>28.55</td>
<td>60.00</td>
<td>81.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Endpoint error</th>
<th>mean</th>
<th>std</th>
<th>R0.5</th>
<th>1.79</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</td>
<td>4.72</td>
<td>5.90</td>
<td>0.49</td>
<td>21.26</td>
<td>0.98</td>
<td>0.80</td>
</tr>
</tbody>
</table>

### Table B.6.: Hydrangea

<table>
<thead>
<tr>
<th></th>
<th>Angular error</th>
<th>Endpoint error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lucas-Kanade</td>
<td>Eigenvector</td>
</tr>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>13.04</td>
<td>13.91</td>
</tr>
<tr>
<td>std</td>
<td>22.31</td>
<td>22.78</td>
</tr>
<tr>
<td>R2.5</td>
<td>57.38</td>
<td>64.00</td>
</tr>
<tr>
<td>R5</td>
<td>38.70</td>
<td>43.90</td>
</tr>
<tr>
<td>R10</td>
<td>26.67</td>
<td>30.22</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>std</th>
<th>mean</th>
<th>std</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0.5</td>
<td>20.39</td>
<td>0.73</td>
<td>4.10</td>
<td>0.49</td>
</tr>
<tr>
<td>R1</td>
<td>12.46</td>
<td>0.78</td>
<td>12.32</td>
<td>0.80</td>
</tr>
<tr>
<td>R2</td>
<td>4.72</td>
<td>0.78</td>
<td>5.90</td>
<td>0.80</td>
</tr>
</tbody>
</table>

### Table B.7.: RubberWhale
### Table B.8: Urban2

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>27.31</td>
<td>29.17</td>
</tr>
<tr>
<td>std</td>
<td>29.81</td>
<td>32.91</td>
</tr>
<tr>
<td>R2.5</td>
<td>88.20</td>
<td>89.34</td>
</tr>
<tr>
<td>R5</td>
<td>77.20</td>
<td>79.58</td>
</tr>
<tr>
<td>R10</td>
<td>63.95</td>
<td>65.74</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>5.50</td>
<td>5.78</td>
</tr>
<tr>
<td>std</td>
<td>7.09</td>
<td>9.15</td>
</tr>
<tr>
<td>R0.5</td>
<td>87.71</td>
<td>89.29</td>
</tr>
<tr>
<td>R1</td>
<td>55.78</td>
<td>55.89</td>
</tr>
<tr>
<td>R2</td>
<td>42.14</td>
<td>40.88</td>
</tr>
</tbody>
</table>

### Table B.9: Urban3

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>28.81</td>
<td>26.12</td>
</tr>
<tr>
<td>std</td>
<td>39.68</td>
<td>38.97</td>
</tr>
<tr>
<td>R2.5</td>
<td>65.54</td>
<td>62.20</td>
</tr>
<tr>
<td>R5</td>
<td>55.81</td>
<td>51.23</td>
</tr>
<tr>
<td>R10</td>
<td>44.98</td>
<td>40.67</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>3.65</td>
<td>3.35</td>
</tr>
<tr>
<td>std</td>
<td>4.15</td>
<td>4.14</td>
</tr>
<tr>
<td>R0.5</td>
<td>68.33</td>
<td>65.79</td>
</tr>
<tr>
<td>R1</td>
<td>60.14</td>
<td>56.73</td>
</tr>
<tr>
<td>R2</td>
<td>50.44</td>
<td>46.56</td>
</tr>
</tbody>
</table>
B. Statistics

<table>
<thead>
<tr>
<th>Angular error</th>
<th>Lucas-Kanade</th>
<th>Eigenvector</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Gradient</td>
<td>Quadrature</td>
</tr>
<tr>
<td>mean</td>
<td>19.45</td>
<td>21.55</td>
</tr>
<tr>
<td>std</td>
<td>32.50</td>
<td>32.56</td>
</tr>
<tr>
<td>R2.5</td>
<td>67.73</td>
<td>74.97</td>
</tr>
<tr>
<td>R5</td>
<td>48.75</td>
<td>55.73</td>
</tr>
<tr>
<td>R10</td>
<td>32.76</td>
<td>39.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Endpoint error</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>1.34</td>
<td>1.53</td>
<td>1.37</td>
<td>1.88</td>
<td>1.93</td>
<td>1.63</td>
</tr>
<tr>
<td>std</td>
<td>1.83</td>
<td>2.01</td>
<td>1.83</td>
<td>2.64</td>
<td>2.91</td>
<td>5.06</td>
</tr>
<tr>
<td>R0.5</td>
<td>42.91</td>
<td>47.54</td>
<td>45.63</td>
<td>57.57</td>
<td>63.60</td>
<td>57.82</td>
</tr>
<tr>
<td>R1</td>
<td>34.01</td>
<td>37.33</td>
<td>34.93</td>
<td>48.85</td>
<td>52.42</td>
<td>43.74</td>
</tr>
<tr>
<td>R2</td>
<td>24.07</td>
<td>25.66</td>
<td>23.33</td>
<td>34.73</td>
<td>35.14</td>
<td>28.01</td>
</tr>
</tbody>
</table>

Table B.10.: Venus

B.2. Leuven

In this section some selected statistics from the Leuven experiment are shown. They are the standard deviation and R10 for AE and the standard deviation and R0.5 for EE.

B.2.1. Apical 2 chamber

Figure B.11.: Angular error standard deviation on A2C
Figure B.12.: Endpoint error standard deviation on A2C

Figure B.13.: Percent of Angular error above 10 on A2C

Figure B.14.: Percent of Endpoint error above 0.5 on A2C
B. Statistics

B.2.2. Apical 3 chamber

Figure B.15.: Angular error standard deviation on A3C

Figure B.16.: Endpoint error standard deviation on A3C
B.2. Leuven

Figure B.17.: Percent of Angular error above 10 on A3C

Figure B.18.: Percent of Endpoint error above 0.5 on A3C
B. Statistics

B.2.3. Apical 4 chamber

Figure B.19.: Angular error standard deviation on A4C

Figure B.20.: Endpoint error standard deviation on A4C
Figure B.21.: Percent of Angular error above 10 on A4C

Figure B.22.: Percent of Endpoint error above 0.5 on A4C
Bibliography


Danielsson, Per-Erik and Olle Seger (1990). “Rotation Invariance in Gradient and Higher Order Derivative Detectors”. In: Computer Vision, Graphics, and Image Processing 49.2, pp. 198–221.


82


Bibliography


