UNIVERSITY OF OSLO
Department of Informatics

The General Flow-Adaptive Filter

With Applications to Ultrasound Image Sequences

Cand Scient Thesis

Are Fritz Charles Jensen

29th October 2003

Preface

This thesis is submitted in partial fulfillment of the requirements for the Candidatus Scientiarum (Cand. Scient.) degree at the Department of Informatics, University of Oslo, Norway. The work started September 2002 and finished October 2003.

The supervisors were Prof. Fritz Albregtsen and Prof. Sverre Holm, both at the Digital Signal Processing and Image Analysis Group within the Department of Informatics. The ultrasound data-sequences were kindly provided by Dr. Andreas Heimdal at GE Vingmed.

A thank you must be handed out to Siv Aina Komegne for last-minute proofreading parts of the text.

Are Fritz Charles Jensen
29th October 2003

Contents

List of Figures

List of Tables

Abstract

While image filtering is limited to two dimensions, the filtering of image *sequences* can utilize three dimensions; two spatial and one temporal. Unfortunately, simple extensions of common two-dimensional filters into three dimensions yield undesirable motion blurring of the images. This thesis addresses this problem and introduces a novel filtering approach termed the general flow-adaptive filter.

Most often a three-dimensional filter can be visualized as a cubic lattice shifted over the data, and at each point the element corresponding to the central coordinate is replaced with a new value based entirely on the values inside the lattice. The general principle of the flow-adaptive approach is to spatially adapt the entire filter lattice to possibly complex spatial movements in the temporal domain by incorporating local flow-field estimates.

Results using the flow-adaptive technique on five filters – the temporal discontinuity filter, a tensor-based adaptive filter, the average, the median and a Gaussian-shaped convolution filter – are presented. Both ultrasound image sequences and synthetic data sets were filtered. An edge-adaptive normalized mean-squared error is used as performance metric on the filtered synthetic sets, and the error is shown to be substantially reduced using the flow-adaptive technique, as much as halved in many instances. There are even indications that simple Gaussian-shaped convolution filters can outperform larger and more complex adaptive filters by implementing the flow-adaptive procedure. For the ultrasound image sequences, the filters adopting the flow-adaptive principles had outputs with less motion blur and sharper contrast compared to the outputs of the non-flow-adaptive filters.

At the cost of flow estimation, the flow-adaptive approach substantially improves the performance of all the filters included in this study.

Chapter 1

Introduction

Ultrasound has long been recognized as a powerful tool for use in the diagnosis and evaluation of many clinical entities. Unfortunately, the ultrasound images are often rather noisy.

Common frame rates for ultrasound scanners are in the range of 30 to 50 frames per second. The visual quality of a running film sequence appears better than the quality of each individual frame. An obvious approach for image enhancement would hence be to let each filtered image-frame embody information from several consecutive frames. Improving the frozen ultrasound images are important since such frames are used for documentation purposes, manual measurements, and automatic object detection and analysis [31, 34]. Of course, an enhancement of the isolated frames also benefits the visual appearance of the video sequence.

Even though the noise in ultrasound images has several components, as discussed in chapter 2, the most prominent is nevertheless characterized by a very low temporal correlation [38]. This implies that temporal smoothing will most often have a substantial impact on the noise level. However, when imaging fast moving objects, e.g., in a cardiac examination, temporal filtering is prone to unwanted motion blurring of the structural edges in the images.

Earlier research

A substantial amount of research has been done on the filtering of ultrasound sequences. Several authors have applied techniques based on the topic of chapter 3, order statistics. Examples on the use of filters based on the median are [9], who used directional median, and [19], who proposed a filter adaptively selecting either the mode or the median. Stack filters were applied by Harvey et. al.[17]. Polynomial fitting, as briefly mentioned in that same chapter, has been tried in [6].

Other alternatives are iterative filters such as the geometric filter[8], and yet another technique is that of using inverse filtering to remove speckle noise [47, 18].

The utilization of higher order moments in adaptively setting filter weights was investigated in [15], and averaging over several heart cycles has also been done [25].

Schistad and Taxt[48] applied a filter based on quantum-mechanical field models, together with an adaptive linear filter and a median filter, on one-dimensional temporal neighborhoods of size 3. The outputs were then filtered spatially with filters from the same classes. They also noted that a simple extension of these filters to three dimensions gave unacceptable motion blur. Evans and Nixon[12] presented a spatio-temporal filter based on a modified two-dimensional least mean squares algorithm, where the iterative weight updates were manipulated both to make the output biased towards the mode or the mean depending on the local speckle pattern, and to reduce motion blur. An overview of several classes of filters and an evaluation based on texture classification can be found in [34].

Goals and contributions of this thesis

Besides giving discussions on ultrasound imaging principles, order statistics based filters and energy-based structure and flow estimation, the main contribution of the thesis is the introduction and evaluation of the general flow-adaptive filter. The general flow-adaptive filter utilizes the low spatial variance of flow fields to simplify and improve filter-adaptability to spatial movement in time sequences. It will be shown that the novel filter approach gives substantial improvement gains in both synthetic sequences and real ultrasound images.

The organisation of this thesis

The text, including this introduction, is organized into nine chapters and three appended sections. Succeeding the present, chapter 2 provides a theoretical summary of the ultrasound image formation process. Chapter 3 presents a broader discussion on the topic of filters based on order statistics, giving a firm basis to discuss the behavior of such filters in the experiments. Two background topics necessary for the appreciation of the conducted experiments; the discontinuity filter and speckle tracking, are described in chapter 4. Local structure and velocity estimation, together with adaptive filtering, using the energy-based approach is discussed in chapter 5. Chapter 6 presents the general flow-adaptive filter. The description of the experimental materials and methods, results and discussion, and the conclusions occupy chapter 7, 8 and 9, respectively. Appendix A holds additional plots, tables and examples of filtered images. A list of mathematical notations is found in appendix B, and appendix C lists excerpts from the program code.

Chapter 2

Ultrasound Imaging Principles

This chapter describes the basic principles of the formation of ultrasound images, and the factors affecting their quality. The material presented is based on Angelsen[2] unless others are referenced.

Ultrasound is a term describing mechanical waves whose frequency is above the range of normal human hearing, which extends from about 30 to 20,000 Hz. In medical ultrasound imaging, frequencies in the range of 2-10 MHz are common for transcutaneous measurements, and for intravascular imaging, tiny catheter-tipped probes operate at frequencies as high as 30 MHz. The ultrasound waves pass through the body and the echoes are registered. By processing the backscattered signals from several directions, an ultrasound image is constructed.

The ultrasound wave signals are generated, and received, by a device called a *transducer*. The transducer, or *array* of transducers, is controlled by a radio frequency (RF) unit allowing the resulting ultrasound beam to be focused and steered. The *scanline processor* extracts the time-domain data acquired by the RF-unit for each pulse, and then the scanline data is further *scan converted* giving the resulting image to be displayed.

2.1 Ultrasound transducer

The transducer probe generates and receives sound waves using a principle called the *piezoelectric effect*. When a voltage source is coupled on each side of a piezoelectric plate, the plate will either expand or contract depending on the polarity of the voltage. Applying an oscillating voltage source makes the plate thickness vibrate, creating the desired mechanical wave.

To obtain the necessary amplitude, the transducer plate must be in resonance, which happens when the plate thickness is half a wavelength. Since the transducer is at resonance, it will keep oscillating a few times after the voltage source

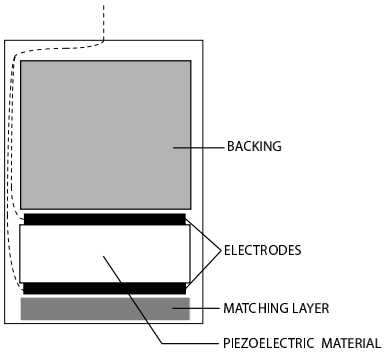


Figure 2.1: Schematic illustration of an ultrasound transducer

has been removed. This is called *the ring down*. As discussed in section 2.3.5, minimizing the pulse length, i.e. maximizing the pulse bandwidth, is crucial for the radial resolution of the image.

There are two common ways to reduce the excessive ringing. One is to mount the transducer plate on an absorbing backing. This dampens the ringing because energy is transmitted into the backing. The disadvantage of this method is that it reduces the amplitude of the pulse as well as the sensitivity of the transducer as a receiver. The other way of reducing the ringing is by using a thin plate between the transducer and the tissue having a stiffness and mass density between that of the transducer and that of the tissue. This facilitates the energy coupling between the transducer and the tissue, making the ringing die out more quickly. The thickness of the plate should be one quarter of the wavelength in the plate, and is often referred to as a *quarter wave impedance transformer*. A schematic illustration of an ultrasound transducer is shown in figure 2.1.

2.1.1 Transducer arrays

By building the transducer using a number of individually controlled smaller elements, the resulting ultrasound beam can be electronically focused and steered by appropriately delaying each element.

There are several different types of arrays. One is the *annular array*, which consists of concentric rings. Such an array cannot be steered, but is able to focus. A *phased array* has a relatively small aperture, and all of the elements are used to form beams emanating from one point. Such arrays are normally used in cardiology applications because the small aperture allows access between the ribs. There are also *linear* and *curved linear arrays* where only parts of the array elements are active at a time. The complexity and cost of the implementation often limits the arrays to about 64-128 elements.

Three-dimensional imaging, where the beam has to be able to steer the azimuthal direction as well, requires the use of two-dimensional arrays. Such arrays usually consist of a very large number of transducers, and it is therefore sought to limit the number of array elements while still obtaining acceptable performance, see [3] and references therein.

2.2 Fundamental acoustics

When an ultrasound pulse encounters a boundary between two tissue structures the pulse will be partially reflected and partially transmitted. The reflection depends on the difference between the *characteristic impedances* of the two materials. The characteristic impedance of a material is given by

$$Z = \rho c$$

where ρ is the mass density and c is the wave propagation velocity of that material. The ratio of the amplitudes of the reflected and the incident pulse is called the *reflection coefficient*, and is, for a beam normal to the interface plane, given by

$$R = \left| \frac{Z_2 - Z_1}{Z_1 + Z_2} \right|$$

Thus, interfaces consisting of materials having a large difference in characteristic impedance reflect a greater fraction of the ultrasound pulse.

The variance of the sound velocity in soft tissue is only about 0.2-0.3% so that the wave propagates approximately like in an homogeneous material. This allows simple calculations to estimate the distance to targets based on the time lag between the pulse transmission and the received reflected pulse. The distance, r, to the target is

$$\tau = \frac{2r}{c} \qquad \Rightarrow \qquad r = \frac{c\tau}{2} \tag{2.1}$$

where τ is the time lag and c is the sound velocity, which is often set to approximately 1540m/s. An important exception to the homogeneity assumption is *fat*, which has both mass density and sound velocity, much lower than that of muscular tissue, causing degradation and artifacts in the image as discussed in section 2.5.

As the pulse propagates through the tissues, the wave intensity is attenuated, which is caused by absorption of wave energy into heat, reflection, scattering and divergence of the ultrasound beam. Because of this attenuation, the reflections from distant targets are much weaker than the reflections from near targets. This is compensated for by using a time-variable amplitude gain, known as *time gain compensation*, or TGC.

2.3 Ultrasound beam

2.3.1 Beam characteristics

The beam radiated from the transducer can be analyzed using *Huygens' principle*, in which each point on the surface acts as a source of a spherical wave. These waves will interfere and generate a beam similar to the one schematically illustrated in figure 2.2(a). Besides the high-intensity mainlobe, there are other, smaller lobes called *sidelobes* that are located around the main lobe. The imaging effects caused by sidelobes are discussed in section 2.3.4 on page 10. The beam is divided into a *nearfield* and a *farfield* region. In the farfield region, also known as the Fraunhofer zone, the wave, due to diffraction, expands with a fixed beamwidth. In the nearfield the beam contracts before it expands, causing an apparent focusing that is referred to as diffraction focusing. The transition between the farfield and the nearfield is not exact, and several different application-dependent criteria are being used. In [2], the distance, d_{nf}, to the nearfield-farfield transition for a plane circular transducer is set to be

$$d_{nf} = \frac{D^2}{2\lambda}$$

where D is the transducer diameter and λ is the wavelength. The 12dB dual-sided beamwidth for the same transducer surface is

$$\theta = \frac{2\lambda}{D}$$

The beam can be focused either electronically, by appropriately delaying array elements, or mechanically as illustrated in figure 2.2(b). The focus will have reduced sharpness due to refraction, and the 12dB focal diameter for a uniformly vibrating disc is

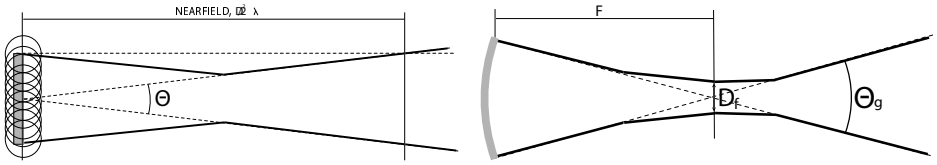
$$D_F = 2\frac{\lambda}{D}F \tag{2.2}$$

where F is the radius of curvature of the disc. For the focusing to have an effect, the geometric focusing angle must be larger than the diffraction beamwidth, and the focus must be in the nearfield of a corresponding non-focused transducer. When using arrays, and the transducer is set to receive, the focus can be varied with time to follow the reflections from the pulse from gradually deeper depths. This rapid updating of the focus during receive is called *dynamic focusing*.

The beam profile becomes more omni-directional as the element size is reduced. Elements as those used in transducer arrays, should therefore be as small as possible, at least less than $\frac{\lambda}{2}$, to ensure the interference necessary for a well-formed steerable beam.

2.3.2 Basic array pattern analysis

In the farfield, the wavefront is approximately planar, and simple models can give insight into array requirements and the formation of mainlobes and sidelobes.



(a) Non-focused transducer (b) Focusing by reshaping the transducer

Figure 2.2: Schematic illustration of the ultrasound beam formation

Let the wave field, $f(\vec{x}, t)$, consist of a monochromatic wave with temporal frequency ω propagating with wavenumber $\vec{k}$,

$$f(\vec{x}, t) = e^{j(\omega t - \vec{k}^t \vec{x})}$$

Placing the array transducers, or sensors, uniformly on the z-axis with an inter-element spacing, d, the wave field measured at the n-th sensor is

$$y_n(t) = e^{j(\omega t - n k_z d)}$$

Letting each of the N sensors be individually weighted before summation yields the total array output:

$$y(t) = \sum_{n=0}^{N-1} w_n e^{j(\omega t - n k_z d)} = e^{j\omega t} \sum_{n=0}^{N-1} w_n e^{-jnk_z d} = e^{j\omega t}\, W(\psi) \qquad (2.3)$$

where $\psi = k_z d$, making $W(\cdot)$ the Fourier transform of the transducer weights. Thus, one can analyze the array output resulting from plane waves arriving from different directions using simple Fourier transform analysis. Figure 2.3 shows the response function of a uniformly weighted array. Note that the *wave equation* limits $W(\psi)$ to represent real propagating signals only when $-\frac{2\pi d}{\lambda} \leq \psi \leq \frac{2\pi d}{\lambda}$, also referred to as the *visible region*[52].

Lobes having the same height as the mainlobe are called *grating lobes* and (2.3) reveals its periodical occurrence whenever $\psi = m2\pi, m \in \mathbb{Z}$. To avoid grating lobes in the visible region, the inter-element distance, d, must satisfy $d < \lambda$. However, steering of the array can be seen as a shift in the beam pattern [52], so to prevent grating lobes to appear in the visible region when the array is allowed to steer $\pm\frac{\pi}{2}$, it is required that $d \leq \frac{\lambda}{2}$. The appearance of grating lobes is equivalent to aliasing in time-series analysis.

Note that the above models assume a continuous wave excitation. With short pulsed waves the sidelobes are blurred, and the clear distinctions between them in the beam pattern are reduced.

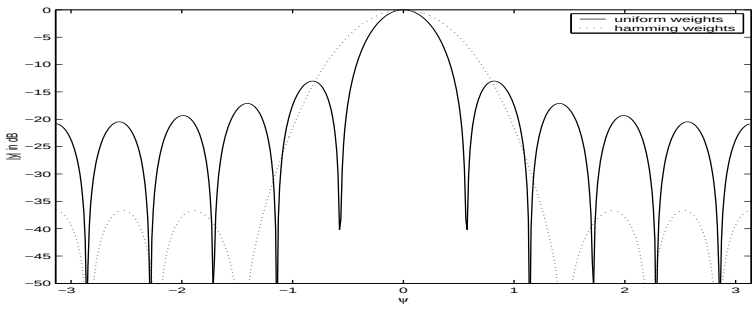


Figure 2.3: Beam pattern of a uniformly spaced, $d = \frac{\lambda}{2}$, linear array with 11 elements using different apodization, or weighting, strategies

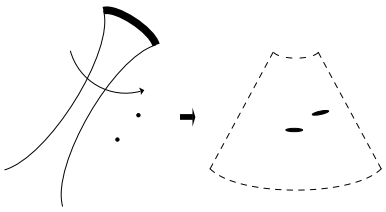


Figure 2.4: How beamwidth affects lateral resolution, from [2]

2.3.3 Beamwidth

The width of the beam limits the lateral resolution as illustrated in figure 2.4. By focusing, the beamwidth can be reduced within a limited distance of the focal point. From (2.2) we see that the focal diameter can be reduced by increasing the frequency while maintaining the transducer diameter, but high-frequency waves attenuate much more than waves of low frequencies which limits the ultrasound penetration depth. Because of this, different frequencies are used when imaging different organs and patients, depending on the depth of penetration needed.

2.3.4 Sidelobes

The sidelobes will spread out in many directions, including outside of the image plane, and result in interfering backscatter degrading the contrast of the image. The sidelobe levels can be reduced by *apodization*, which in the case of a continuous aperture means that the transducer is built in such a way that its center vibrates with a higher amplitude than its periphery. When using a transducer array, the amplitudes can be individually manipulated mimicking the same behavior as that of the continuous apodization, or be based on well-known time-analysis weighting strategies like *Bartlett*, *Blackman* or *Hamming*, see [36]. Figure 2.3 shows the effect of apodization using the Hamming window in the farfield model described in

section 2.3.2. Reduction of the sidelobes is done at a cost of a wider mainlobe, but the perceived gain in contrast resolution is often higher than the corresponding loss in lateral resolution.

Since a transducer array consists of discrete elements, the sidelobe levels are higher than that of a corresponding continuous aperture. Reducing the size and increasing the number of the elements mitigates this, and improves the effect of apodization.

2.3.5 Pulse bandwidth

The length of the pulse determines the radial, or range, resolution of the imaging system. Targets that are radially close cannot be resolved if the distance between them is less than half the pulse length. In this case the reflections overlap, and separation is impossible. The least number of times the transducer oscillates is practically constant, which makes it possible to reduce the pulse length by increasing the frequency.

2.4 Scattering and speckle

When the ultrasound beam encounters a reflecting object that is small compared to the ultrasound wavelength, a fraction of the beam will scatter in all directions. This can benefit imaging, since tissue interfaces usually have a certain roughness which scatters the beam, making interfaces visible even though the beam is not perpendicular to the tissue surface.

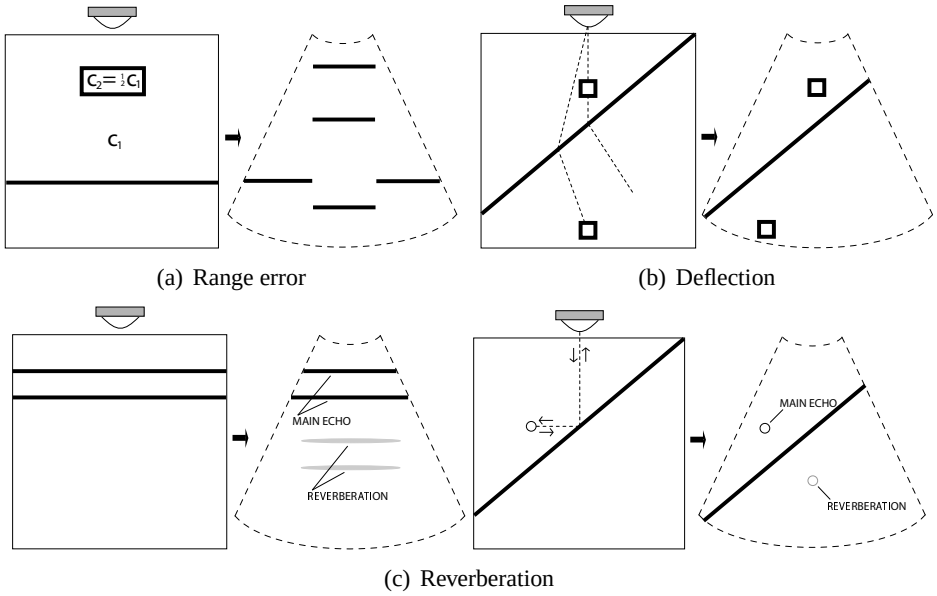
When several of these sub-resolution scatterers interact with each other they cause intricate interference patterns called *speckle*. Speckle is seen as grainy or mottled texture with a rapid spatial variation, and might often be mistaken for real structure. The pattern depends on the frequency of the transmitted pulse and the beamwidth, and will, in theory, not change with lateral or radial displacement. This latter can be utilized in *speckle tracking*, where blood flow and tissue motion is estimated by template-matching algorithms, as described in section 4.2.

2.5 Tissue inhomogeneity

Tissue inhomogeneity cause irregular velocity and mass variance along the beam path which can cause distortion and degradation of the image. Several of the resulting effects are discussed here.

2.5.1 Ranging errors

The time of echo arrival is converted into range using (2.1), which assumes a constant and predefined wave velocity. When the actual velocities deviate from this



(a) Range error (b) Deflection

(c) Reverberation

Figure 2.5: Imaging errors caused by tissue inhomogeneity, from [2]

value, image distortions can appear, as illustrated in figure 2.5(a). An important example is fat, through which the beam has a velocity of about 1450m/s which is considerably lower than the average sound velocity in tissue of about 1540m/s.

2.5.2 Deflection

Refraction occurs whenever an incident beam strikes an interface at a non-perpendicular angle, and can cause error in the location of the imaged object. This is often referred to as *deflection error*. Figure 2.5(b) illustrates how the system assumes a straight beam path, while in reality it is bent due to the refraction caused by the difference in sound velocity in the two tissues.

2.5.3 Wave front aberration

Irregular variations in the wave velocity also destroy the wave front in a process called *wave front aberration*. The resulting beam is diffuse, and this causes a degradation of the spatial resolution.

2.5.4 Shadowing and enhancement

When the ultrasound pulse propagates through an object with a high level of attenuation, the area distal to the object receives a signal with reduced amplitude. This is known as *shadowing* and causes weaker reflections from the distal area, which

is typically seen as dark streaking behind the highly attenuating objects. The opposite effect is known as *enhancement*, and gives high brightness levels distal to objects having a low level of attenuation compared to the surroundings, see [44] and references therein. Note that a reduction, or removal, of the enhancement and shadowing artifacts is not always desired, since they often contain valuable information for the clinician [45].

2.5.5 Reverberation

As the reflected pulses encounters interfaces on their way back, they will again be reflected and sent back deeper into the tissue. Such multiple reflections are known as *reverberation*, and is illustrated in figure 2.5(c). Even though these multi-reflected pulses are weaker, they can still cause severe degradation of the image. Especially the low mechanical impedance of fat compared to muscular tissue, and the difference in mechanical impedance between the transducer face and the tissue, give rise to strong reverberation artifacts.

2.6 Temporal resolution

The time it takes to generate an image depends on the number of beams and the depth of interest. Each beam needs $2R/c + T_0$ time to collect data from depth R, where T_0 is an extra wait period to make sure that the pulse is sufficiently attenuated. This gives, for an image consisting of N beams, the frame rate:

$$frame\,rate = \frac{1}{N(2R/c + T_0)}$$

Thus, there is a tradeoff between the number of beams, imaging depth and temporal resolution. When using 128 beams and a depth of about 16cm, a frame rate of about 30 frames per second is achievable.

Each frame can be constructed from several sweeps of the beam with different focal depths to improve the lateral resolution. This is referred to as *composite transmit focus*, and the frame rate is reduced by a factor equal to the number of focal depths. The severe reduction of the frame rate renders this method unsuitable for imaging fast moving structures like the heart, but it is fully applicable for imaging internal organs in the abdomen.

Chapter 3

Adaptive Filters Based on Order Statistics

This chapter presents selected topics in the field of order-statistics based filtering.

Using low-pass linear filters as noise suppressors have the undesirable side-effect of blurring sharp transitions in the input signal. The nonlinear filters presented here try to overcome this weakness while retaining the noise suppressing effect.

Filtering is performed by letting a window move over the data sequence, and at each point the data values within the window, $\vec{x} = (x_1, \ldots, x_N)$, are used as input to the filter. The output of the filter, y_i, replaces the value in the signal at the position corresponding to the window center.

The *ordered* input sequence to the filter will be denoted $\vec{x}_{()}$, that is,

$$\vec{x}_{()} = (x_{(1)}, \ldots, x_{(N)}), \qquad x_{(1)} \leq x_{(2)} \leq \ldots \leq x_{(N)}$$

3.1 Median filters

Letting the output of the filter being the *median* of the $N = 2m + 1$ input values, i.e.,

$$y_i = x_{(m)} \tag{3.1}$$

we obtain what is referred to as the *median filter*[41]. If N is an even number, the median is defined to be the average of $x_{(\lceil N/2 \rceil)}$ and $x_{(\lfloor N/2 \rfloor)}$ [37], but the focus of this text will be on the case of N being odd.

3.1.1 Characteristics of the median filter

By studying figure 3.1 on the next page it appears that the median filter has two very desirable properties: It preserves sharp edges, or step functions, and it is well

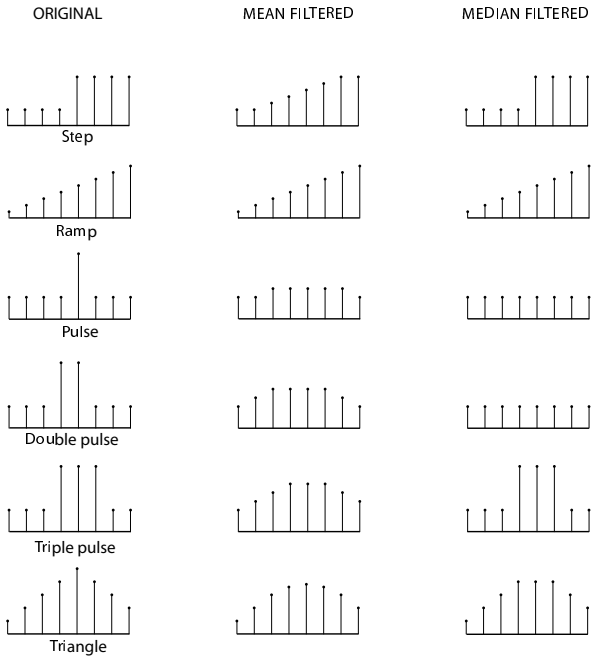


Figure 3.1: Median filtering of one-dimensional example signals

suited for impulse noise filtering.

Edge information is very important for human perception [49]. The edge preserving qualities of the median filter makes it more suitable for edge filtering compared to its low-pass linear counterparts, which do poorly on the, by definition, high frequency contents of edges.

Pulse functions, whose periods are less than one-half the window length, will be suppressed. In a wide-band sensor, short impulses might represent valid signal pulses which will be filtered out. Thus, in some cases the median filter will provide noise suppression, while in others it will cause signal suppression.

Filter shapes In the multi-dimensional case, various forms of filter windows can be used. Figure 3.2 on the facing page illustrates some of the two-dimensional window shapes often encountered. A window shape which is symmetric around the origin, and includes the origin, will preserve sharp edges in any direction, although the square, and circle ring, windows will result in only slight edge alterations [23].

Signal borders If the input signal is of finite extent, there will be signal borders where only parts of the applied filter window is defined. There are two approaches

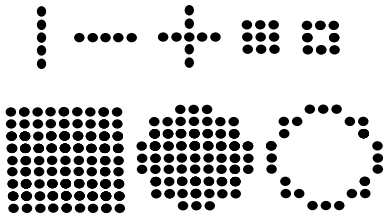


Figure 3.2: Common two-dimensional filter windows

to solve this problem [41]: In the first, the filter window is truncated so that only well defined samples, i.e. samples on the inside of the signal borders, are included in the calculation of the median. In the second approach, the sequence is appended with the sufficient amount of samples, and the usual (3.1) is applied.

Computational concerns The most intuitive way of implementing the median filter is to sort all the elements, and then pick the one in the middle. Any which way the sorting is done, one still has a worst-case lower bound of $O(N \log N)$, where N is the length of the input sequence [53]. Trying to reduce the complexity by using *heap*-like structures to avoid sorting the entire array will generally not prevail. However, due to the fact that the samples inside the filter window is only partly replaced as the window is moved across the signal, sorting improvements beyond the above mentioned restrictions can be achieved [21]. Special hardware-oriented algorithms have also been proposed [10].

3.1.2 Noise reduction

By assuming certain noise density distributions, variance formulas yielding quantitative information about the noise suppression of the median filters can be found. The focus here will be on white noise, i.e., noise sequences being independent identically distributed. Due to the nonlinearity of the median filters, it is not possible to separate the signal effects and noise effects as simply as for linear filters. Thus, the easiest case of a constant signal is assumed.

The signal sequence consists of elements modeled as

$$x = m + z$$

where $E[z] = 0$ and $E[x] = m$. Let $F(x)$ and $f(x) = F'(x)$ denote the distribution and density functions of the x variables, and σ_x denote the variance. The density of the median of a sequence $(x_1, x_2, \ldots, x_N)$ is [46]:

$$g(y) = N \binom{N-1}{(N-1)/2} f(y) F(y)^{(N-1)/2} [1 - F(y)]^{(N-1)/2}. \qquad (3.2)$$

This distribution is for large N approximately normal [23], $N(\tilde{m}, \sigma_N)$, where $\tilde{m}$ is the continuous median, i.e., $F(\tilde{m}) = 0.5$, and

$$\sigma_N^2 = \frac{1}{N4f^2(\tilde{m})} \approx Var[median\ of\ (x_1, x_2, \ldots, x_N)]. \tag{3.3}$$

By replacing the factor $1/N$ in (3.3) with $1/(N+b)$, where $b = 1/[4f^2(\tilde{m})\sigma_x^2] - 1$, the equation (3.3) becomes exact for $N = 1$, thereby giving a better approximation for small N.

Uniform distribution Letting the x variables be uniformly distributed on $[0, 1]$, one can compute the variance of the median exactly using (3.2):

$$Var[median\ of\ (x_1, x_2, \ldots, x_N)] = \frac{1}{4(N+2)} = \frac{\sigma_x^2}{N+2} \cdot 3.$$

Normal distribution When the x variables are normally distributed, (3.2) must be computed numerically. Alternatively we can get an approximation using (3.3). Including the modifications for small N, it yields

$$Var[median\ of\ (x_1, x_2, \ldots, x_N)] \approx \frac{\sigma_x^2}{N + \pi/2 - 1} \cdot \frac{\pi}{2}, \qquad N = 1, 3, 5, \ldots$$

Comparing this result with the variance of the *average* of $(x_1, x_2, \ldots, x_N)$, which is σ_x^2/N, it is evident that the average filter is much better at suppressing normal white noise than the median filter. Noting that the average is actually the maximum likelihood estimate of m, this result comes hardly as a surprise.

Double exponential distribution If, on the other hand, the x variables have a double exponential distribution, i.e.,

$$f(x) = \frac{\sqrt{2}}{\sigma_x} e^{-\sqrt{2}|x-m|/\sigma_x}$$

we get, using (3.3):

$$Var[median\ of\ (x_1, x_2, \ldots, x_N)] \approx \frac{\sigma_x^2}{(N - 1/2)} \cdot \frac{1}{2}$$

which is about 50% smaller than the variance, σ_x^2/N, of the average. Here, the median is the maximum likelihood estimate of m, which is obvious when one recalls that the minimum of

$$\sum_{i=1}^{N} |x_i - a|$$

is attained for $a = median\ of\ (x_1, x_2, \ldots, x_N)$.

Error rate p	$n = 3$	$n = 9$	$n = 25$
0.01	0.00030	0	0
0.05	0.00725	0.000033	0
0.1	0.028	0.00089	0.0000002
0.2	0.104	0.0196	0.00037
0.3	0.216	0.099	0.0175
0.4	0.352	0.267	0.1538
0.5	0.5	0.5	0.5

Table 3.1: Probabilities of erroneous reconstruction in case of impulse noise

Impulse noise By making the assumptions that the distorting impulses, or spikes, occurs at each sample with equal probability, p, and that the impulses are all of the same value, d, we get

$$x_i = \begin{cases} d & \text{with probability } p \\ s_i & \text{with probability } (1-p) \end{cases}$$

where $s_i \neq d$ denotes the value of the original signal, which is assumed approximately constant in the local window neighborhood.

The output value, y_i, will be correct if, and only if, the number of errors within the local window neighborhood is less than half the total number of elements in that same neighborhood. The number of erroneous points in the neighborhood has a binomial distribution, yielding the following result:

$$\begin{aligned} P(\text{correct reconstruction}) &= P(y_i = s_i) \\ &= \textstyle\sum_{k=0}^{(N-1)/2} \binom{N}{k} p^k (1-p)^{N-k} \end{aligned}$$

A few values of the probability of erroneous reconstruction are shown in table 3.1.

A general conclusion of the above examples is that medians are better than means when the distributions are heavily tailed.

3.1.3 Frequency response

The frequency response of the median filter for general signals, which can be seen as linear combinations of complex exponentials, does not exist due to the nonlinearity of the filter. In this subsection the power spectrum distribution of a median filtered *single* cosine wave is derived. We focus on the simplest case in which $N = 3$, giving:

$$x(t) = \cos(\omega t), \qquad t \in \Re, \quad 0 \leq \omega \leq \pi$$

$$y(t) = median\ of [x(t-1), x(t), x(t+1)], \qquad t \in \Re$$

A few minutes of staring at the above equations verifies that

$$y(t) = \begin{cases} \cos(\omega(t-1)) & 0 < t \leq 1/2 \\ \cos(\omega t) & 1/2 < t \leq T/2 - 1/2 \\ \cos(\omega(t+1)) & T/2 - 1/2 < t \leq T/2 \end{cases}$$

where $T = 2\pi/\omega$, i.e., $y(t)$ is periodic with period T.

By expanding $y(t)$ as a Fourier series with coefficients

$$c_k = \frac{1}{T}\int_0^T e^{\mathrm{j}tk\omega}, \qquad k = 0, \pm 1, \pm 2, \ldots$$

we can extract the components having frequency $\pm\omega$, i.e., when $k = \pm 1$. By representing $cos(t)$ in its complex exponential form, the calculations needed are quite simple, leading to

$$c_1 = c_{(-1)} = \frac{1}{2}\Big(1 - \frac{\omega}{\pi} + \frac{\omega}{\pi}\cos w\Big), \qquad 0 \leq \omega \leq \pi$$

giving the spectral effect at frequencies $\pm\omega$ equal to

$$c_1^2 + c_{(-1)}^2 = 2c_1^2.$$

One can also compute the total spectral effect, i.e., the variance of $y(t)$:

$$\sigma_y^2 = \frac{1}{T}\int_0^T y^2(t)dt = \frac{1}{2}\Big[1 + \frac{1}{\pi}(\sin(2\omega) - 2\sin\omega)\Big], \qquad 0 \leq \omega \leq \pi$$

Figure 3.3 on the facing page shows σ_y^2, $2c_1^2$ and also the variance of a corresponding three-point average filter, σ_a^2. Note that $2c_1^2 = \sigma_y^2$ for linear filters. For $w < 2\pi/3$ the plot demonstrates the low-pass characteristics of the median filter, showing a similar response as the corresponding average filter. For higher frequencies the similarities break down, and we see that both σ_y^2 and $2c_1^2$ has the same response for $\omega = \pi$ and $\omega = 0$. This latter observation is explained by the fact that the median filter will preserve the form of a sequence $x = (\ldots, 1, -1, 1, -1, \ldots)$, although shifted one step.

For larger N, analytic solutions to the above equations are infeasible, and numerical integration must be used. Frequency response of non-simple signals has been attempted computed empirically as quotients of Fourier transform of output and input signals, but the result is obviously dependent on the input signals chosen [23].

3.1.4 Some alternatives to the standard median filter

Some commonly encountered alternatives to the standard median filter are discussed briefly below. Richer descriptions can be found in [41].

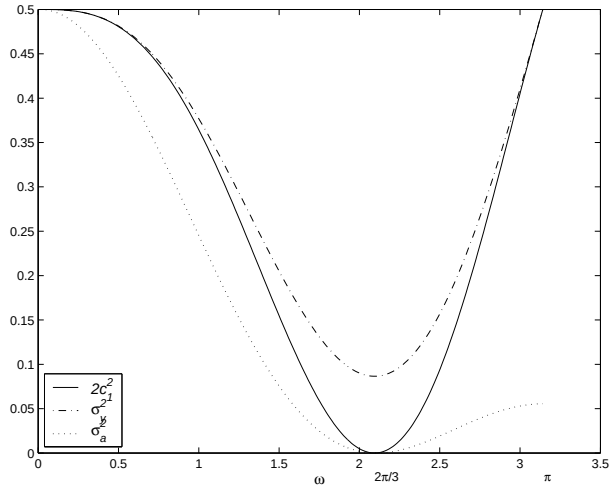


Figure 3.3: Filtering of cosine wave. Spectral effect.

Separable In multiple dimensions, the separable median filter is the result of several successive one-dimensional median filtering operations. In the two-dimensional case, letting $x_{i,j}$ denote the image value at row i and column j, and letting $y_{i,j}$ denote the corresponding filter output, we get

$$y_{i,j} = median\ of(z_{i,j-v}, z_{i,j-v+1}, \ldots, z_{i,j+v-1}, z_{i,j+v})$$
$$z_{i,j} = median\ of(x_{i-v,j}, x_{i-v+1,j}, \ldots, x_{i+v-1,j}, x_{i+v,j})$$

where $N = 2v + 1$ is the one-dimensional filter-window length. The separable median filter of length N has greater output variance than its non-separable $N \times N$ counterpart [41]. The main advantage of the separable median filter is its low computational complexity. Sorting two sequences of length N is more efficient than sorting one sequence of length $N \times N$, which is evident from our discussion on computational concerns in section 3.1.1 on page 17.

Recursive In the recursive median filter, already computed output samples are used in the calculation of the output y_i:

$$y_i = median\ of(y_{i-v}, y_{i-v+1}, \ldots, x_i, \ldots, x_{i+v-1}, x_{i+v})$$

Its output tends to be more correlated than that of the standard median filter, and has more immunity to impulsive noise [41]. The recursive median filter can also be made separable by obvious modifications.

Weighted The *weighted median* is the estimator, a, that minimize [41]:

$$\sum_{i=1}^{N} w_i |x_i - a|$$

By letting $w \diamond x$ denote the duplication of x w times, we can describe the filter by:

$$y_i = median\ of(w_{-v} \diamond x_{i-v}, \ldots, w_v \diamond x_{i+v})$$

By properly choosing the filter weights, time information can be incorporated in the filter, e.g., weighting the central pixel more heavily. *Kth nearest neighbor median filter* and *two-dimensional in-place growing filter*[5] are two examples of filters obtained by proper weighting.

Multistage The standard median filter performs poorly on multi-dimensional signals with a high level of fine details. For example, in images, thin lines and sharp edges are not preserved. Such details are, as mentioned above, very important to the human perceptual system, and consequently standard median filtering can cause severe visual degradation. Thus, several efforts have been made to take into account structural information. One such attempt gives us the *multistage* median filter:

$$\begin{aligned}
y_{i,j} &= median\ of(median\ of(z_1, z_2, x_{i,j}), median\ of(z_3, z_4, x_{i,j}), x_{i,j}) \\
z_1 &= median\ of(x_{i,j-v}, \ldots, x_{i,j}, \ldots, x_{i,j+v}) \\
z_2 &= median\ of(x_{i-v,j}, \ldots, x_{i,j}, \ldots, x_{i+v,j}) \\
z_3 &= median\ of(x_{i+v,j-v}, \ldots, x_{i,j}, \ldots, x_{i-v,j+v}) \\
z_4 &= median\ of(x_{i-v,j-v}, \ldots, x_{i,j}, \ldots, x_{i+v,j+v})
\end{aligned}$$

Multistage median filers can preserve details in horizontal, vertical and diagonal directions, due to the corresponding sub-filters.

3.2 L-filters

An important generalization of the median filter is the *L-filter*[4]:

$$y_i = \sum_{j=1}^{N} \alpha_j x_{(j)} \tag{3.4}$$

where N is the size of the filter, $x_{()}$ are the ordered window samples, and $\alpha_j, j = 1, \ldots, N$, are weight coefficients.

By using the weight coefficients

$$\alpha_j = \begin{cases} 1 & \text{if } j = (N+1)/2 \\ 0 & \text{otherwise} \end{cases}$$

one obtains the standard median filter, and by using the weight coefficients $\alpha_j = 1/N$ one gets the standard running average filter. Setting all the coefficients to zero except for $\alpha_i = 1$ the i*th rank order operation*[4] is obtained. Obvious modifications leads to the *max/min filter*[43]. The α*-trimmed mean filter*[41] is also obtainable by properly setting the filter coefficients.

3.2.1 Adaption to different noise distributions

The filter coefficients can be chosen to satisfy an optimality criterion that is related to the probability distribution of the input noise. Considering a constant signal corrupted by zero-mean white noise, we can model the output as

$$x_i = s + n_i$$

where x_i is the observed output, s is the constant signal, and n_i are independent identically distributed random variables satisfying $E[n_i] = 0$. Assuming that the noise distribution is symmetric, the condition that $s = E[y_i]$ is satisfied by imposing the constraint:

$$\sum_{j=1}^{N} \alpha_j = \vec{\alpha}^t \vec{1} = 1 \tag{3.5}$$

where $\vec{\alpha} = (\alpha_1, \ldots, \alpha_N)$ and $\vec{1} = (1, \ldots, 1)$. Letting $\vec{x}_{()} = (x_{(1)}, \ldots, x_{(N)})$ and $\vec{n}_{()} = (n_{(1)}, \ldots, n_{(N)})$ the mean squared error norm is given by

$$\begin{aligned} \epsilon^2 &= E\Big[(y_i - s)^2\Big] = E\Big[(\vec{\alpha}^t \vec{x}_{()} - s)^2\Big] \\ &= E\Big[(\vec{\alpha}^t (s\vec{1} + \vec{n}_{()}) - s)^2\Big] = E\Big[(\vec{\alpha}^t \vec{n}_{()})^2\Big] \\ &= \vec{\alpha}^t R \vec{\alpha} \end{aligned} \tag{3.6}$$

where $R = E[\vec{n}_{()} \vec{n}_{()}^t]$. Minimizing the above function with the constraint (3.5) can be done using Lagrange multipliers. The Lagrangian function is given by

$$\Phi(\vec{\alpha}) = \vec{\alpha}^t R \vec{\alpha} + \lambda(\vec{\alpha}^t \vec{1} - 1)$$

Setting the derivatives with regards to $\vec{\alpha}$ equal to zero gives, assuming R is non-singular,

$$\frac{\partial \Phi}{\partial \vec{\alpha}} = R\vec{\alpha} + \lambda \vec{1} = 0 \quad \Rightarrow \quad \vec{\alpha} = -R^{-1}\lambda \vec{1} \tag{3.7}$$

and using (3.5) gives

$$\lambda = \frac{-1}{\vec{1}^t R^{-1} \vec{1}}$$

Plugging this back into (3.7) yields

$$\vec{\alpha} = \frac{R^{-1}\vec{1}}{\vec{1}^t R^{-1} \vec{1}} \tag{3.8}$$

Thus, having the noise correlation matrix, R, one can easily obtain the filter coefficients using (3.8). A more general design scheme for applications involving non-constant known signals is given in [4].

3.2.1.1 Computation of the correlation matrix

Evaluation of the R in (3.8) requires expressions for the marginal and the bivariate densities of $n_{(j)}$. Denoting the parent distribution and density of the noise as $F_n()$ and $f_n()$, respectively, the density of $n_{(i)}, i = 1, \ldots, N$ is given by [46]:

$$g_{n_{(i)}}(x) = \frac{N!}{(i-1)!(N-i)!} f_n(x) F_n(x)^{i-1} [1 - F_n(x)]^{N-i}.$$

The joint density of $n_{(i)}$ and $n_{(j)}, i, j = 1, \ldots, N$ $(i < j)$ is [4]:

$$g_{n_{(i)}n_{(j)}}(x, y) = K_{i,j} f_n(x) f_n(y) F_n(x)^{i-1} [F_n(y) - F_n(x)]^{j-i-1} [1 - F_n(y)]^{N-j}$$

where

$$K_{i,j} = \frac{N!}{(i-1)!(j-i-1)!(N-j)!}$$

The symmetric correlation matrix R is obtained by integration:

$$R_{i,j} = \begin{cases} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} xy g_{n_{(i)}n_{(j)}}(x, y) dx dy & \text{if } i < j \\ \int_{-\infty}^{\infty} x^2 g_{n_{(i)}}(x) dx & \text{if } i = j \end{cases}$$

The complexity of these equations makes numerical integration generally necessary, even for simple parent distributions of the original noise.

The resulting optimal coefficients for several noise distributions, and for $N = 3, 9, 25$, is found in [4]. The results for the uniform and normal distributions are their corresponding maximum likelihood estimators, i.e., the midpoint, $\alpha_1 = \alpha_N = 1/2$ for the uniform case, and the average, $\alpha_j = 1/N, j = 1, \ldots, N$, for the normal distribution case. Generally, the results confirm the statement in section 3.1.2 on page 19 in that the weights located in the center becomes more pronounced as the noise distribution grows heavier tailed.

3.2.1.2 Using empirical data

If the desired filter-output at each sample is known, the minimization of

$$\epsilon^2 = E\big[(y - d)^2\big] \tag{3.9}$$

where y is the filter output and d the desired output, is obtained by the filter coefficients [40]:

$$\vec{\alpha} = E[\vec{x_{()}} \vec{x_{()}}^t]^{-1} E[d\vec{x_{()}}] \tag{3.10}$$

By explicitly indexing the M input-samples, (3.10) can be written as

$$\vec{\alpha} = \left[\frac{1}{M} \sum_{i=1}^{M} [\vec{x^i_{()}} \vec{x^i_{()}}^t] \right]^{-1} \cdot \frac{1}{M} \sum_{i=1}^{M} [d_i \vec{x^i_{()}}]$$

In an on-line, or sample-by-sample estimate, the updates would be of the *least mean square*[11] type:

$$\vec{\alpha}(k+1) = \vec{\alpha}(k) + \Delta \vec{x}_{()}^{f(k)}(d_{f(k)} - \vec{\alpha}^t(k)\vec{x}_{()}^{f(k)})$$

where $k = 1, 2, \ldots,$ $f(k)$ is indexing the samples in a circular fashion, and Δ is a properly chosen step-size.

Perception-related cost functions Palmieri and Croteau[40] introduce in addition a factor, k, in (3.9), yielding a modified mean squared error function,

$$\epsilon_k^2 = E[k(y-d)^2]$$

where k_i is a feature factor signaling the importance of a close fit at sample i. The feature extractor could be an edge detector or generally an image-dependent parameter that reflects the relevance of that specific image area to good image perception. The resulting filter coefficients are:

$$\vec{\alpha}_k = E[k\vec{x_{()}}\vec{x_{()}}^t]^{-1}E[kd\vec{x_{()}}]$$

3.2.2 L*l*-filters

While the L-filter operates on the *ordered* input, $x_{()}$, losing spatial information, the linear FIR filter operates in the spatial domain, not utilizing order information. The *Ll-filter*[39] is a generalized L-filter combining information both before and after ranking.

The output of a linear filter can be written

$$y_l = \vec{\beta}^t\vec{x}$$

where $\vec{\beta}$ contains the filter weights, and $\vec{x} = (x_1, x_2, \ldots, x_N)^t$ is the window sample values. The corresponding L-filter can be written

$$y_L = \vec{\alpha}^t P(\vec{x})\vec{x}$$

where $\vec{\alpha}$ contains the filter weights, and $P(\vec{x})$ is a $N \times N$ permutation matrix sorting the elements of $\vec{x}$ in ascending order.

To account for both arrangements of the linear and L-filter, one would need N^2 coefficients. Namely, a data sample in the window is multiplied by a different coefficient according to its position both before and after ranking. A simplified version of the estimator that needs only $2N$ coefficients is what is called the L*l*-filter and is given by [39]:

$$y = \vec{\alpha}^t P(\vec{x})B\vec{x} \quad or\ equivalently \quad y = \vec{\beta}^t P^t(\vec{x})A\vec{x}_{()}$$

where $B = diagonal(\beta_1, \ldots, \beta_N)$ and $A = diagonal(\alpha_1, \ldots, \alpha_N)$.

The mean square error surface is generally a non-convex function. However, observing that the function becomes convex if either $\vec{\alpha}$ or $\vec{\beta}$ is held fixed, one can reach a solution using bilinear parameterization: Fix $\vec{\alpha}$, then find the best value for $\vec{\beta}$, then fix $\vec{\beta}$ to the new value and optimize $\vec{\alpha}$, and so on. The on-line updates can be written [39]:

$$\vec{\beta}(k+1) = \vec{\beta}(k) + \Delta_\beta \vec{x}_\alpha(k)(d_{f(k)} - \vec{\beta}^t(k)\vec{x}_\alpha(k))$$
$$\vec{\alpha}(k+1) = \vec{\alpha}(k) + \Delta_\alpha \vec{x}_\beta(k)(d_{f(k)} - \vec{\alpha}^t(k)\vec{x}_\beta(k))$$

$$\vec{x}_\alpha(k) = P^t(\vec{x}^{f(k)})A(k)\vec{x}_{()}^{f(x)}$$
$$\vec{x}_\beta(k) = P(\vec{x}^{f(k)})B(k+1)\vec{x}^{f(x)}$$

where Δ_α and Δ_β are step parameters. Bilinear parameterization procedures generally converge to local minima, but using several random starting parameters mitigates the problem [11].

3.3 Stack filters

All the rank order operators, median filters, weighted median filters and weighted order statistics filters are part of a broader class of filters known as *stack filters*[54]. These filters share two properties: the *threshold decomposition* property and an ordering property known as the *stacking* property.

3.3.1 The threshold decomposition and stacking property

The threshold decomposition property lets us decompose the M-valued input signal into a set of $M-1$ binary signals, and then filter each signal independently with its own binary filter, before summing the outputs for the final result. The k-th binary signal, where $k \in \{1, 2, \ldots, M-1\}$, is obtained by thresholding the input signal at the value k. Figure 3.4 on the next page illustrates the threshold decomposition architecture on a median filter of size three.

Letting the input window samples be $\vec{x}$ and the filter output be y_i, the process can be described by

$$y_i = \sum_{k=1}^{M-1} f(T_k(\vec{x}))$$

where

$$T_k(\vec{x}) = \Big(T_k(x_1), T_k(x_2), \ldots, T_k(x_{N-1}), T_k(x_N)\Big)$$

in which

$$T_k(x_i) = \begin{cases} 1 & \text{if } x_i \geq k \\ 0 & \text{else} \end{cases}$$

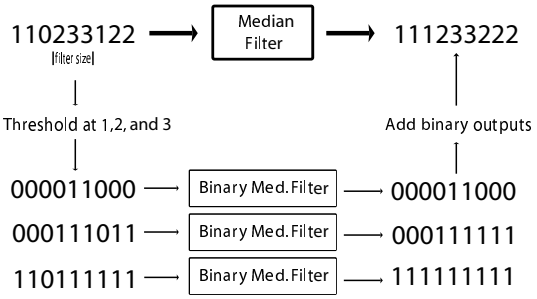


Figure 3.4: Median filtering by threshold decomposition

and $f(\cdot)$ is the boolean filter.

The other requirement defining stack filters is the stacking property, which says that whenever the boolean function on level l outputs a 1, then the boolean functions on every level below level l must also output 1's. From this property, and the requirement that the same boolean function is used on all levels, it follows that only *positive* boolean functions are allowed [54]. A boolean function, f, is positive if, and only if

$$f(\vec{a}) \leq f(\vec{b}) \quad \text{whenever} \quad \vec{a} \leq \vec{b}$$

where $\vec{a} \leq \vec{b}$ denotes that if $a_i = 1$ it implies that $b_i = 1$ for all i.

The above requirements makes stack filters very suitable for hardware implementation. The decomposition property makes it highly parallel, and the stacking property facilitates the summation of the binary filter-outputs by allowing simple binary search.

Input compression Coyle et al.[22] use the fact that the output of a stack filter is always one of the sample points in its input window in making an alternative implementation of the stack filter using fewer levels in the decomposition. Which sample is chosen depends on the relative ranks and positions of the samples in the window, not on the actual magnitude of the samples. The sample points in a window of size N can therefore be mapped to the integers 1 through N before the stack filter is applied. Once this compressed data has been filtered, the compressed sample chosen is then mapped back to its original value. The number of binary filters needed is reduced at the cost of sorting the input samples, and the mapping involved.

3.3.2 Stack filter design

Manually designing the boolean function of the stack filter is of course possible in some applications, and in [54] the entire set of the 20 possible filters of length

3 is listed and discussed. However, computation of optimal boolean functions is most often done using a truth table. Any boolean function, f, of N variables can be represented by a length 2^N truth table. The truth table may consist of a decision vector, $\vec{d} = (d_0, d_1, \ldots, d_{2^N-1})$, where $d_i = f(\vec{x_i})$ and $\vec{x_i}$ is one of the binary sequences which correspond to the window arrays with N elements. Thus, the problem of finding an optimal boolean function is equivalent to finding the corresponding decision vector.

3.3.2.1 Mean absolute error

The shear volume of positive boolean functions for $N > 5$ makes exhaustive search infeasible. Fortunately, the properties of the stack filter together with the mean absolute error criteria has led to several useful algorithms.

The mean absolute error between the output of the filter, $f(\cdot)$, and some desired signal, s, given the corrupted window signal, $\vec{x}$, is [32]:

$$\begin{aligned} MAE_f &= E[|s - f(\vec{x})|] = E\left[\left| \sum_{k=1}^{M-1} [T_k(s) - f(T_k(\vec{x}))] \right|\right] \\ &= \sum_{k=1}^{M-1} E[|T_k(s) - f(T_k(\vec{x}))|] \end{aligned}$$

The second equality comes from the threshold decomposition property and the third equality is a result of the stacking property. Thus, the overall mean absolute error is simply the sum of the mean absolute errors on each of the threshold decomposition levels.

3.3.2.2 Linear program

The optimal filtering problem over the class of stack filters under the mean absolute error criterion can be formulated as the following linear program [32]:

$$minimize \quad \sum_{i=0}^{2^N-1} C_i \cdot d_i, \text{ subject to the constraints:}$$

$$d_i \leq d_j \text{ if } \alpha_i \leq \alpha_j, \qquad 0 \leq d_j \leq 1 \text{ for all } j$$

where each coefficient C_i depends on the joint statistics of the corrupted and the desired signal. The problem with this program is that the constraints on f implied by the stacking property grows exponentially with the window size, and that the joint statistics coefficients are seldom known. These limitations gave rise to adaptive filters based on training sequences.

3.3.2.3 Training sequences

Lin and Coyle[33] developed an adaptive algorithm based on training sequences. In this algorithm, each decision variable, d_i, can take on values in the interval $[0, 1]$ with resolution $1/N$, where N is some positive integer. In this case, d_i can be

interpreted as the probability that the filter outputs a 1 when x_i is the input. In the actual implementation, the decision variables are rescaled to integer values in the interval $[-N/2, N/2]$. A vector of strictly hard decisions can be recovered at any time by thresholding the soft decision vector at zero.

The algorithm, which is shown to converge, starts by setting all the decision variables to zero, and for each threshold layer and for each input sample, do:

1. Find the correct decision variable, d_i from the input. If the correct output is 1 then $d_i = d_i + 1$, else $d_i = d_i - 1$

2. If the stacking constraint is not satisfied for d_i, iterate and swap d_i with decisions violating the constraint until the constraint is fulfilled.

Faster adaptive algorithms have been developed [57, 56]. However, the principles of the new algorithms remain the same, and only the frequency of which the stacking-constraint check is performed, and the techniques of which it is enforced, is altered.

In image filter design, algorithms can incorporate perceptual information by means of using the *weighted mean absolute error* criterion, weighting errors in perceptual-critical parts heavier [20]. Obtaining the decision vector using genetic programming algorithms has also been explored [7].

3.4 Adaptive space-varying filters

The filters described in the previous sections are usually optimized for å specific type of noise and sometimes a specific type of signal. However, this is not usually the case, and in many applications the signal characteristics vary considerably within the signal. In images, for example, neighborhoods of edges are characterized by large signal variance as opposed to neighborhoods located in flat regions. Space-varying filters are trying to incorporate information about the local signal neighborhood into the filtering process.

3.4.1 Varying filter sizes

One simple way of adapting the filtering process to the local signal characteristics is to vary the filter size based on estimation of local signal variance [41].

An example could be to start with a rather large window at each point. Then, if the estimated signal variance is above a certain threshold, the window size is reduced. If the new local signal variance is still above a certain threshold, the window size is again reduced, and so on, until a window of size 3×3 is reached.

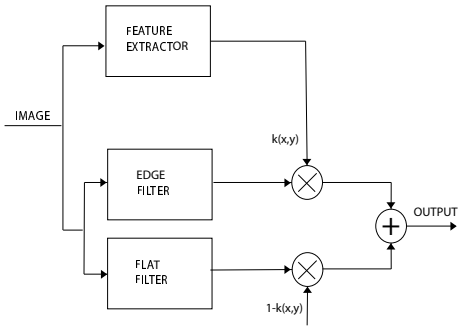


Figure 3.5: Simple space-varying filter using a linear combination of two filters, from [40]

3.4.2 Linear combinations of multiple isotropic filters

Another way of obtaining local adaptability is by combining outputs from several different filters, each optimized for different signal characteristics. The weighting of the output from the different filters can be based on feature extractors tuned to the target signal characteristics of the specific filter [40].

Figure 3.5 shows a block diagram of a simple adaptive image filter differentiating edges and flat neighborhoods. The feature extractor is an edge detector giving an output between 0 and 1 depending on how "edge-like" the local neighborhood is deemed to be.

3.4.3 Anisotropic window adaption

The filter window itself can be made adaptable to different signal structures. In images, for example, the filter window can be aligned with the edge, so as not to encompass pixels from both sides of the edge.

One way of estimating local structure is that of Granlund and Knutsson[14] which is explained in chapter 5. Other examples include fitting multi-variable polygons to the signal [31], and techniques based on binary morphological erosion and dilation [41]. More pixel-based filters, like the α-trimmed mean, the sigma filter and the k-nearest neighbors, can be argued belong to another class of filters. Descriptions of those filters can be found in most introductory books on image processing.

Chapter 4

Discontinuity Filter and Speckle Tracking

This chapter presents two important topics referred to later in this thesis. The temporal discontinuity filter is used extensively throughout the experiments, and speckle tracking is used in addition to the energy-based method for estimating motion in the ultrasound image sequences.

4.1 Discontinuity filter

4.1.1 Introduction

This section describes the temporal filtering method for ultrasound image sequences suggested by Olstad[38]. The general idea is that the time evolution for each individual spatial coordinate is divided into *homogeneous* regions, whereupon the temporal filtering is performed with minimal interaction across the boundaries defining the regions.

A synthetic example of a time sequence divided into five homogeneous regions for a fixed spatial coordinate is shown in figure 4.1. The sequence could have been generated by a high-intensity object present in the time interval $[t_0, t_1]$ and a rapidly moving structure present in the interval $[t_2, t_3]$. Reconstruction based on piecewise constant and linear functions are superimposed on the intensity sequence.

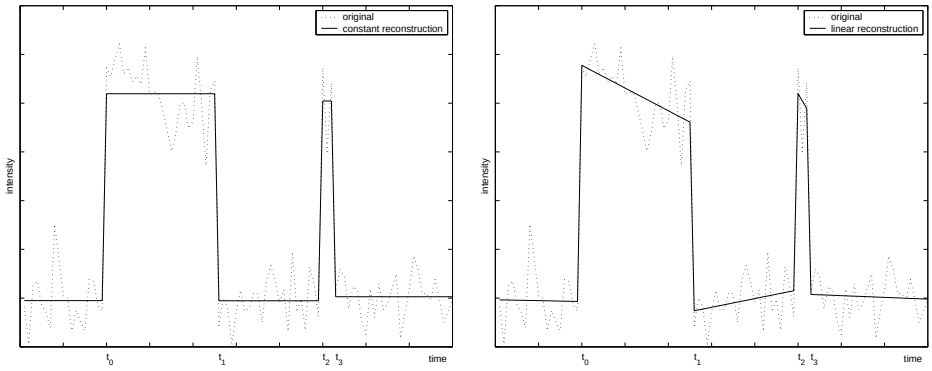
4.1.2 Homogeneity measure requirements and discontinuity detection

Let $I = (x_1, x_2, \ldots, x_N)$ be any sequence of temporal consecutive intensity values. A homogeneity measure for I is any nonnegative function:

$$e: \quad I \longrightarrow \Re^+$$

satisfying

$$e(I) \geq e(I_1) + e(I_2)$$



(a) Constant reconstruction (b) Linear reconstruction

Figure 4.1: Optimal discontinuity locations and piecewise reconstruction

whenever $I = I_1 \cup I_2$ and $I_1 \cap I_2 = \emptyset$.

Let α_M contain a set of M ascending indices $t_i : 0 < t_0 < t_1 < \ldots < t_{M-1} < N$ indicating discontinuities in a time window of N samples. The discontinuities generate the following $M + 1$ subsequences:

$$\begin{aligned} I_0 &= (x_0, \ldots, x_{t_0-1}) \\ I_1 &= (x_{t_0}, \ldots, x_{t_1-1}) \\ &\vdots \\ I_M &= (x_{t_{M-1}}, \ldots, x_{N-1}) \end{aligned}$$

An error function, E, on the set of discontinuities, α_M, can then be defined using the homogeneity measure on each I_j:

$$E(\alpha_M) = \sum_{j=0}^{M} e(I_j) \tag{4.1}$$

Since the set of all possible α_M is finite, a solution to minimizing (4.1) will always exist, but it is not necessarily unique. An arbitrary choice is made in the case of multiple minimums. The integer value M is an algorithm parameter, and depends on a-priori information and computational complexity considerations.

4.1.3 Filtering and homogeneity measures

4.1.3.1 Piecewise constant functions

A natural choice of homogeneity measure is the square error,

$$e(\ (x_1, x_2, \ldots, x_K)\ ) = \sum_{i=1}^{K} (x_i - \overline{x})^2$$

where $\overline{x}$ denotes the mean value of $x_1, \ldots, x_K$. Thus, $E(\cdot)$ measures how well the intensity function can be approximated by a function of $M+1$ constant segments. An example of intensity reconstruction using optimal constant segments is shown in figure 4.1(a). This approach will have substantial impact on the noise level, and increase the sharpness of the intensity transitions. Efficient algorithms for the computation of the optimal discontinuity set utilizing dynamic programming can be found in [38]

4.1.3.2 Piecewise linear functions

Reconstruction using a piecewise linear function can be obtained by using the homogeneity measure:

$$e(\ (x_1, x_2, \ldots, x_K)\ ) = \min_{a,b} \sum_{i=1}^{K} \Big(x_i - (a + b \cdot i)\Big)^2$$

This makes $E(\cdot)$ measure how well the intensity function can be approximated by a function of $M+1$ *linear* segments, as illustrated in figure 4.1(b). Higher order parametric functions, such as splines, can also be utilized in the reconstruction [38].

4.1.3.3 Incorporating spatial persistence

Components modeling spatial persistence can be incorporated by weighting the intensity observations in the error measurement , i.e.,

$$e(\ (x_1, x_2, \ldots, x_K)\ ) = \sum_{i=1}^{K} \gamma_i \cdot (x_i - \overline{x})^2$$

where γ_i is a weight coefficient reflecting the spatial persistence of x_i. γ_i could for example be based on the order statistic of x_i in a local spatial neighborhood such that $\gamma_i = 1$ if x_i is the median and dropping towards zero as x_i turns to the minimum or maximum value.

4.1.4 Shorter time windows

When using shorter time windows, ordering the window samples in a cyclic manner, as illustrated in figure 4.2, improves performance. The two discontinuities are computed such that the homogeneity measures of the two intervals in the cyclic ordering is minimized.

Allowing a small *leakage* across the boundaries, i.e., including weighted border pixels in the reconstruction, will improve the noise suppression at the cost of only a minor edge degradation.

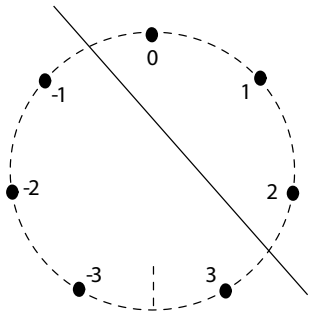


Figure 4.2: Window samples ordered in a cyclic manner

4.2 Speckle tracking

Speckle tracking is a method used to estimate blood flow and structural movement in ultrasound image sequences based on template-matching algorithms utilizing properties of the speckle patterns found in such images. The desired properties are that speckle patterns have a high spatial variance, and that, as discussed in section 2.4, the speckle pattern will, theoretically, not change with lateral or radial displacement. Of course, rotation, compression, phase aberration, and movement out of the scan plane all change the patterns.

Figure 4.3 illustrates the template-matching algorithm. First a small kernel region, X, is taken from one image frame. Then the kernel is used as a template to search for a match in a larger region, Y, of the subsequent image. The translation coordinates of the *best* match is taken as the local flow estimate. Three common criteria for finding the best match are: correlation,

$$\epsilon_{m,n} = \sum_{i,j}(X_{i,j} \cdot Y_{i+m,j+n})$$

sum-absolute difference, or SAD,

$$\epsilon_{m,n} = \sum_{i,j}|X_{i,j} - Y_{i+m,j+n}|$$

and normalized correlation,

$$\rho_{m,n} = \frac{\sum_{i,j}\left[(X_{i,j} - \overline{X}) \cdot (Y_{i+m,j+n} - \overline{Y})\right]}{\sqrt{\sum_{i,j}(X_{i,j} - \overline{X})^2 \cdot \sum_{i,j}(Y_{i+m,j+n} - \overline{Y})^2}}$$

where $\overline{X}$ and $\overline{Y}$ are the mean of X and Y, respectively, the sums are taken over the entire kernel, and m, n are the translations.

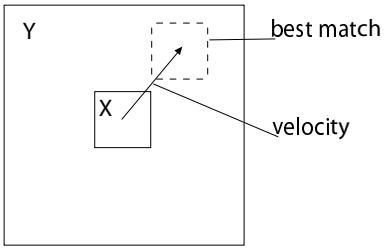


Figure 4.3: Illustration of the two-dimensional template-matching algorithm

The first criterion, the correlation, has the worst performance, while the two others perform quite equally, with a slightly better normalized correlation [13]. The failure of the correlation metric is obvious, at least for small kernels, since it is highly susceptible to variance in speckle amplitude. The main reason for the performance difference between the other two is that the sum-absolute difference does not compensate for changes in mean or variance, as the computationally more costly normalized correlation does. Note that in all such template-matching procedures, there is a tradeoff between kernel size and the ability to independently detect the motion of small structures.

An efficient method for computing the normalized correlation can be found in [51], who also introduces cost functions limiting the resulting movement variation. [30] is a survey on general medical image registration.

Chapter 5

Energy-Based Local Structure and Velocity Estimation

This chapter presents a way of estimating, and representing, local structure and velocity in multi-dimensional signals.

Fundamental to the following approach, is the recognition of the importance of local gradient directions, and its sufficiency in determining velocity. In the human visual system this importance has been demonstrated by both physiological and psychological studies [49]. It will further be assumed that, generally, the spatial variation of the gradient direction will be much slower than the spatial variation of the image itself, i.e., the local signal can be thought of as one-dimensional.

By studying the local Fourier transforms, interesting properties surface. If the signal is simple, that is, approximately one-dimensional, the energy distribution in the Fourier domain is concentrated along a narrow sector in the gradient direction. The less variation of the local orientation the narrower the sector will be, as illustrated in figure 5.1. The distribution of energy along the radial direction of this sector will reflect the frequency properties of the local neighborhood in the gradient direction.

A description of the local neighborhood, in terms of both direction and frequency, can thus be obtained by partitioning the local Fourier transform and studying the energy contribution from the different parts.

5.1 Phase-independent energy estimation

This section introduces the concept of analytic signals, and their use in obtaining phase-independent energy estimation.

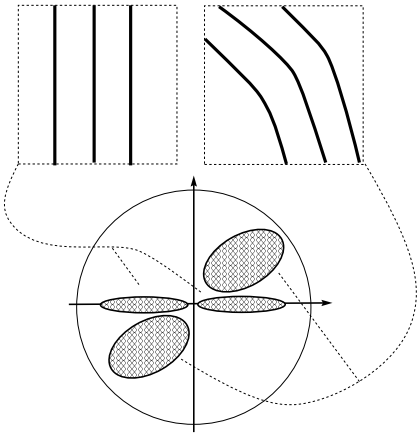


Figure 5.1: Two neighborhoods and their corresponding Fourier energy distributions

5.1.1 Incentive for phase independence

As a way of introducing the essence of phase independence, consider the simple real signal f given by

$$f(x) = A\cos\omega x \qquad A > 0.$$

where ω reflects the angular frequency. Measurements of the instantaneous value of f at some point $x = x_0$ will result in any value between -A and A, depending on x_0, i.e., we cannot obtain the amplitude, A, in any simple way.

5.1.2 The analytic signal

To overcome this limitation we would like to have also at our disposal a $\pm\frac{\pi}{2}$ phase-shifted (positive $\frac{\pi}{2}$ shift for the positive frequencies, and negative shift for the negative frequencies) version of the original signal;

$$h(x) = -A\sin\omega x.$$

By combining the two real signals into a complex representation we get

$$f_A(x) = A[\cos\omega x + i\sin\omega x] = Ae^{i\omega x}$$

where the amplitude, A, is given directly by

$$A = |f_A(x)| = \sqrt{[f(x)]^2 + [h(x)]^2}.$$

Thus, what is needed is a way of filtering the original signal to obtain the $\pm\frac{\pi}{2}$ phase-shifted version. In the Fourier representation this corresponds to multiplication

with the imaginary unit i and changing the sign for negative frequencies. The resulting filter

$$F_{Hi}(u) = \begin{cases} i & \text{if } u > 0 \\ -i & \text{otherwise} \end{cases}$$

is known as an *ideal Hilbert transform*[36].

By combining the original signal with the Hilbert-transformed version, f_{Hi}, in the following way

$$f_A = f - if_{Hi}$$

we obtain what is referred to as an *analytic signal*[36]. In the Fourier domain we can represent this, by using the linearity of the Fourier transform, as

$$\begin{aligned} F_A(u) &= F(u) - iF_{Hi}(u) \\ &= F(u) - i[F(u) \cdot i\text{sign}(u)] \\ &= 2F(u) \cdot \text{step}(u) \end{aligned}$$

where $\text{step}(u)$ is unity where $u > 0$ and 0 elsewhere, and F_A, F and F_{Hi} are the Fourier transforms of f_A, f and f_{Hi}, respectively. We see that the desired, analytic signal, corresponding to f, is obtained by suppressing all the negative frequencies and multiply by two.

5.1.3 Analytic signals in multiple dimensions

Until now we have focused on one-dimensional signals. A simple generalization of the Hilbert transform to multiple dimensions is not possible since the concept of positive and negative frequencies is not defined in that case. As suggested in [14] the concept of positive and negative frequencies can be employed by introducing a direction of reference in the Fourier domain. Given a directional vector, $\vec{e}$, we can label a frequency coordinate, $\vec{u}$, positive if $\vec{u}^t\vec{e} > 0$ and negative if $\vec{u}^t\vec{e} < 0$.

The Hilbert transform can thus in the multiple domain be defined, given a reference vector $\vec{e}$, as

$$F_{Hi}(\vec{u}) = F(\vec{u}) \cdot i\text{sign}_{\vec{e}}(\vec{u}),$$

where

$$\text{sign}_{\vec{e}}(\vec{u}) = \begin{cases} 1 & \text{if } \vec{u}^t\vec{e} > 0 \\ -1 & \text{if } \vec{u}^t\vec{e} < 0 \end{cases}$$

Given the Hilbert transform above, the multi-dimensional analytic signal is defined in the same way as for one-dimensional signals; $f_A = f - if_{Hi}$.

5.1.4 Band-pass filter

By filtering the analytic signal with a band-pass filter, we can obtain a phase-independent estimation of the energy distribution in the band-pass region. The resulting phase-independent filter will, in the Fourier domain, thus be real and have

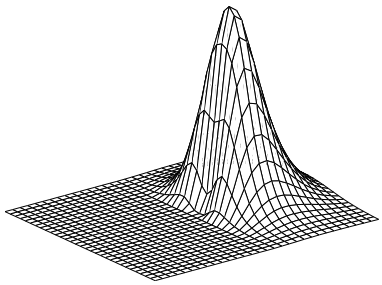


Figure 5.2: Fourier representation of a 2D phase-independent band-pass filter

zero in the negative frequencies, see fig 5.2.

Since real signals will always be Hermitian [14], the only way of obtaining a filter with zero negative frequencies, is by letting the filter coefficients be complex. The actual spatio-temporal convolution filter pairs comprising of the real and imaginary part of the complex filter are obtained by methods explained in section 5.7 on page 51.

5.2 Tensor representation of local structure

Using vectors as a way of representing orientation of local structure, like lines and planes, has an innate ambiguity problem. A 180° rotation of a line or a plane leads to no change at all. Thus, two lines being of nearly the same angle may end up having vector representations at opposite directions, leading to an unacceptable distance measure. Another problem of using vectors is that of certainty, both in representation and its use in averaging. When estimating how well a neighborhood fits the local one-dimensionality assumption, one can use the *double-angle* representation [26] in the 2D case, thereby making the norm of the vector signal certainty. However, in higher dimensional space the double-angle technique obviously breaks down. The inconvenience of using vectors when averaging flow-fields is addressed in section 5.5 on page 47.

5.2.1 The outer-product tensor

Knutsson[27] introduced an alternative way of representing local structure using outer-product tensors. As we shall see, using tensors exceeds the above mentioned shortcomings.

Let $\hat{\vec{x}}$ be a unit vector in the direction of maximal local variance. The tensor representation, $\mathbf{T}$, is given by

$$\mathbf{T} \equiv A\hat{\vec{x}}\hat{\vec{x}}^t \tag{5.1}$$

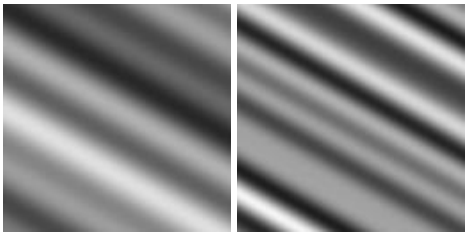


Figure 5.3: Local simple neighborhoods with equal orientation but differing signals

where A can be any constant greater than zero.

5.2.2 Orientation tensor requirements

As mentioned above, one of the main incentives for using tensors is the requirement that both the orientation vectors $\vec{x}$ and $-\vec{x}$ yields the same result. From the definition of the tensor (5.1) we immediately see that this requirement is met.

It is also necessary that the tensor representing orientation be invariant to the type of signal actually encountered, i.e. obtaining the same angle independent of amplitude, or whether it is an edge, line etc. As an example, the orientation obtained from the two signals in figure 5.3 should be the same. Again, from the definition of the tensor, which has no reference to the contents of the signal, we see that this requirement is met. The actual orientation estimation is shown in section 5.3.

The orientation tensor should also locally preserve the angle metric, i.e., the tensor, **T**, should change proportionally to the change in local orientation. The proof that this holds is omitted here, but can be found in [27].

5.3 Orientation estimation

The general idea of energy-based direction estimation is to probe the Fourier space in several directions with filters each picking up energy in a particular angular sector, and then combining the filter outputs yielding the direction of most signal variance. This section shows how a combination of outputs from specific filters can be used to estimate the orientation tensor presented in section 5.2.1 on the facing page.

5.3.1 The filter sets

Due to the angular invariance required by the orientation tensor, it is immediately recognized that the filters must be distributed equally over the entire orientation space. Note that the phase independent filter outputs will be symmetric so that the

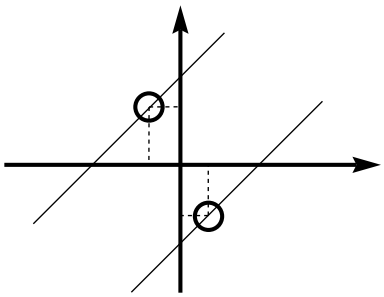


Figure 5.4: Relative contribution from two filters

output of $\vec{x}$ and $-\vec{x}$ gives the same result. In the 2D case it may seem at first that the obvious number of filters is two, but as indicated in figure 5.4 there will be no way of differentiating contributions from the filters (bold lines) along opposing directions of the two thin lines. Two particular areas in the Fourier plane giving identical contributions from the two filters are indicated by circles.

Thus, the minimum number of filters required for the 2D orientation space is three. Distributing them equally gives the orienting vectors:

$$\begin{array}{rcl} \hat{\vec{n}}_1 & = & (1,0)^t \\ \hat{\vec{n}}_2 & = & (a,b)^t \\ \hat{\vec{n}}_3 & = & (-a,b)^t \end{array} \tag{5.2}$$

where $a = 0.5$ and $b = \sqrt{3}/2$, as also illustrated in figure 5.5 on the next page.

By following the same reasoning one can show that the minimum number of filters in three-dimensions is six, and their orienting vectors are:

$$\begin{array}{rclrcl} \hat{\vec{n}}_1 & = & c(a,0,b)^t & \hat{\vec{n}}_2 & = & c(-a,0,b)^t \\ \hat{\vec{n}}_3 & = & c(b,a,0)^t & \hat{\vec{n}}_4 & = & c(b,-a,0)^t \\ \hat{\vec{n}}_5 & = & c(0,b,a)^t & \hat{\vec{n}}_6 & = & c(0,b,-a)^t \end{array}$$

where $a = 2$, $b = (1 + \sqrt{5})$ and $c = (10 + 2\sqrt{5})^{-1/2}$.

5.3.2 Filter outputs

In the following the analysis is restricted to real valued *simple* neighborhoods, i.e., neighborhoods that can be expressed as

$$s(\vec{\xi}) = g(\vec{\xi} \cdot \hat{\vec{x}}) \tag{5.3}$$

where s and g are real functions. Two-dimensional examples of such signals are given in figure 5.3

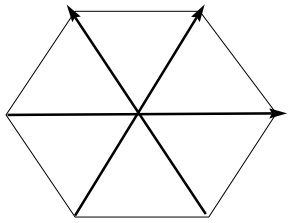


Figure 5.5: Two-dimensional filter orientations

As shown in section 5.1 on page 37 the filters used in the phase-independent energy estimation are zero over one half of the Fourier space, also known as *quadrature filters*[26]. Given the filter direction vector, $\hat{\vec{n}}_k$, we have

$$F_k(\vec{u}) = 0 \qquad \text{if } \vec{u} \cdot \hat{\vec{n}}_k \leq 0.$$

The output $\mathbf{q}_k$ of the corresponding filter will, as mentioned above, be complex. The magnitude $q_k = |\mathbf{q}_k|$ will be independent of whether the local signal is an edge or line. If g is a sinusoidal signal, the magnitude will be completely phase invariant, as demonstrated in section 5.1.

The Fourier transform of the simple signal $s(\vec{\xi})$ can be expressed as

$$S(\vec{u}) = G(\vec{u} \cdot \hat{\vec{x}})\delta^{line}_{\hat{\vec{x}}}(\vec{u})$$

where $\delta^{line}_{\hat{\vec{x}}}(\vec{u})$ is an impulse-line in the direction of $\hat{\vec{x}}$.

Let the filter functions be spherically separable, i.e.,

$$F(\vec{u}) = R(\rho)D(\hat{\vec{u}}), \qquad \rho = |\vec{u}|.$$

The filtering of S by F results in:

$$\mathbf{q} = \int_{-\infty}^{\infty} F(\vec{u})S(\vec{u})d\vec{u} = \int_{-\infty}^{\infty} R(\rho)D(\hat{\vec{u}})G(\vec{u} \cdot \hat{\vec{x}})\delta^{line}_{\hat{\vec{x}}}(\vec{u})d\vec{u}$$

replacing $\vec{u} = \rho\hat{\vec{x}}$ yields:

$$\begin{aligned} \mathbf{q} &= D(\hat{\vec{x}}) \textstyle\int_0^{\infty} R(\rho)G(\rho)d\rho + D(-\hat{\vec{x}}) \int_0^{\infty} R(\rho)G(-\rho)d\rho \\ &= \mathbf{a}D(\hat{\vec{x}}) + \mathbf{a}^*D(-\hat{\vec{x}}) \end{aligned}$$

where

$$\mathbf{a} = \int_0^{\infty} R(\rho)G(\rho)d\rho$$

and $\mathbf{a}^*$ denotes complex conjugate. The complex conjugate comes from the Hermitian property of the Fourier transform, $G(\rho)$, of the real signal $s(\vec{\xi})$.
Since the filter is zero over one half of the Fourier domain, either $D(\hat{\vec{x}})$ or $D(-\hat{\vec{x}})$ will be zero. This makes it possible to write the filter output as

$$q = A[D(\hat{\vec{x}}) + D(-\hat{\vec{x}})] \tag{5.4}$$

where $A = |\mathbf{a}|$ can be seen as the local signal amplitude.

As first suggested by Knutsson[26] the following directional function will be shown to have quite desirable properties:

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

where $\hat{\vec{n}}_k$ is the filter orienting vector.

Combining equation (5.4) and (5.5) gives the filter output magnitude

$$q_k = A(\hat{\vec{u}} \cdot \hat{\vec{n}}_k)^2 \tag{5.6}$$

where A is independent of the filter orientation and depends only on the radial distribution of the signal spectrum, $G(\rho)$, and the filter function, $R(\rho)$.

The radial function, $R(\rho)$, can be chosen arbitrarily, and is typically some kind of band-pass function. When actually implementing the filters the usual limitations of filter design must be considered. The function suggested by Knutsson[14] is given by

$$R(\rho) = e^{-\frac{4}{B^2 \ln 2} \ln^2(\rho/\rho_i)}$$

where B is the relative band-width and ρ_i is the center frequency. Figure 5.2 shows the ideal filter reference function with $B = 2$ and $\rho_i = \pi \cdot 2^{-1.5}$.

5.3.3 Tensor construction

Here it is shown that, given a simple neighborhood, the orientation tensor can be estimated exactly by linearly combining the result of the quadrature filters. In non-simple neighborhoods the resulting tensor still conveys interesting properties, as explained in section 5.4 on page 46. Due to notational simplicity, we focus on the two-dimensional construction, but the three-dimensional case is done in exactly the same manner.

Define the tensors $\hat{\mathbf{N}}_k$ as being the outer product of the orienting vectors $\hat{\vec{n}}_k$, i.e.,

$$\hat{\mathbf{N}}_k = \hat{\vec{n}}_k \hat{\vec{n}}_k^t$$

Letting the signal orienting vector be:

$$\vec{x} = (x_1, x_2)^t$$

gives, by using the equations (5.2) and (5.6), the following filter output magnitudes:

$$\begin{array}{rcl} q_1 & = & Ax^{-2}x_1^2 \\ q_2 & = & Ax^{-2}(a^2x_1^2 + 2abx_1x_2 + b^2x_2^2) \\ q_3 & = & Ax^{-2}(a^2x_1^2 - 2abx_1x_2 + b^2x_2^2) \end{array}$$

where $x = |\vec{x}|$. Combining the tensors, $\hat{\mathbf{N}}_k$, by their corresponding output magnitudes, q_k, we obtain the tensor

$$\mathbf{T}' = \sum_k q_k \hat{\mathbf{N}}_k$$

which has the following components:

$$\begin{array}{rcl} t'_{11} & = & \frac{3}{4}A(x_1^2x^{-2} + \frac{1}{2}) \\ t'_{22} & = & \frac{3}{4}A(x_2^2x^{-2} + \frac{1}{2}) \\ t'_{12} = t'_{21} & = & \frac{3}{4}Ax_1x_2x^{-2} \end{array}$$

If we subtract $\frac{3}{8}A$ from the diagonals we end up with a tensor of the desired form (5.1):

$$\mathbf{T}' - \frac{3}{8}A\mathbf{I} = \frac{3}{4}A\hat{\vec{x}}\hat{\vec{x}}^t$$

Using the fact that the sum of all the filter outputs, q_k, is $\frac{3}{2}A$, we can form new tensors

$$\mathbf{M}_k = \frac{4}{3}\hat{\mathbf{N}}_k - \frac{1}{3}\mathbf{I}$$

making $\mathbf{T}$ a linear combination of the $\mathbf{M}_k$ tensors, i.e.,

$$\mathbf{T} = A\hat{\vec{x}}\hat{\vec{x}}^t = \sum_k q_k(\frac{4}{3}\hat{\mathbf{N}}_k - \frac{1}{3}\mathbf{I})$$

$$\Downarrow$$

$$\mathbf{T} = \sum_k q_k \mathbf{M}_k. \tag{5.7}$$

Thus, the orientation tensor can be estimated by filtering the signal with the appropriate set of quadrature filters, and using the filter output magnitudes as weights in equation (5.7). Note that the $\mathbf{M}_k$ tensors do not depend on the signal, and can thus be precalculated.

As mentioned above, the method of obtaining the three-dimensional results is identical, and the $\mathbf{M}_k$ tensors in equation (5.7) will for the three-dimensional case be [14]:

$$\mathbf{M}_k = \frac{5}{4}\hat{\mathbf{N}}_k - \frac{1}{4}\mathbf{I}.$$

5.4 Interpretation of the orientation tensor

Acquired data does not tend to be simple, in the sense of equation (5.3). However, it is still possible to find a best approximation to the estimated tensor, $\mathbf{T}$, corresponding to a simple neighborhood, $\mathbf{T}_s$, i.e., minimizing

$$\Delta = |\mathbf{T} - \mathbf{T}_s|, \qquad \mathbf{T}_s = A\hat{\vec{x}}\hat{\vec{x}}^t$$

where subtraction is done component-wise and the norm of a tensor is given by:

$$|\mathbf{T}|^2 = \sum_{ij} t_{ij}^2 = \sum_n \lambda_n^2$$

where λ_n are the eigenvalues of $\mathbf{T}$. The tensor, $\mathbf{T}_s$, minimizing Δ is:

$$\mathbf{T}_s = \lambda_1 \hat{\vec{e}}_1 \hat{\vec{e}}_1^t$$

where λ_1 are the largest eigenvalue, and $\hat{\vec{e}}_1$ is its corresponding eigenvector. The value Δ indicates how well the one-dimensional hypothesis fits the acquired data; the smaller the value the better the fit.

As illustrated in figure 5.6, one can in three-dimensional space categorize the local neighborhood into a spatially *planar*, a *linear*, or an *isotropic* case depending on the eigenvalue-distribution of $\mathbf{T}$:

1. **Planar case:** $\mathbf{T} \simeq \lambda_1 \hat{\vec{e}}_1 \hat{\vec{e}}_1^t \qquad (\lambda_1 \gg \lambda_2 \simeq \lambda_3)$
   This corresponds to a simple neighborhood, i.e., the spatial variation is in only one direction, given by $\hat{\vec{e}}_1$.

2. **Linear case:** $\mathbf{T} \simeq \lambda_1(\hat{\vec{e}}_1 \hat{\vec{e}}_1^t + \hat{\vec{e}}_2 \hat{\vec{e}}_2^t) \quad (\lambda_1 \simeq \lambda_2 \gg \lambda_3)$
   This corresponds to a neighborhood that is approximately constant on *lines*. The direction of the lines is given by $\hat{\vec{e}}_3$.

3. **Isotropic case:** $\mathbf{T} \simeq \lambda_1(\hat{\vec{e}}_1 \hat{\vec{e}}_1^t + \hat{\vec{e}}_2 \hat{\vec{e}}_2^t + \hat{\vec{e}}_3 \hat{\vec{e}}_3^t) \quad (\lambda_1 \simeq \lambda_2 \simeq \lambda_3)$
   This corresponds to where the energy is distributed in no particular direction.

Relative strength of the eigenvalues The eigenvalues linearly reflect the amount of band-pass energy in the direction of the corresponding eigenvectors.

By modeling non-simple neighborhoods by combining simple neighborhoods oriented in perpendicular directions, equation (5.7), for three dimensions, becomes

$$\mathbf{T} = \sum_k \left(q_{k_1} \mathbf{M}_{k_1} + q_{k_2} \mathbf{M}_{k_2} + q_{k_3} \mathbf{M}_{k_3} \right) = A_1 \hat{\vec{x}_1} \hat{\vec{x}_1}^t + A_2 \hat{\vec{x}_2} \hat{\vec{x}_2}^t + A_3 \hat{\vec{x}_3} \hat{\vec{x}_3}^t$$

where q_{k_i}, A_i and $\vec{x_i}$ are the output of filter k, the radial energy, and the direction of the i-th simple neighborhood, respectively. Since $\hat{\vec{x}_1}$, $\hat{\vec{x}_2}$ and $\hat{\vec{x}_3}$ are all perpendicular to each other, we immediately recognize that $\hat{\vec{x}_1}$, $\hat{\vec{x}_2}$ and $\hat{\vec{x}_3}$ will be the eigenvectors of $\mathbf{T}$, with A_1, A_2 and A_3 as their corresponding eigenvalues.

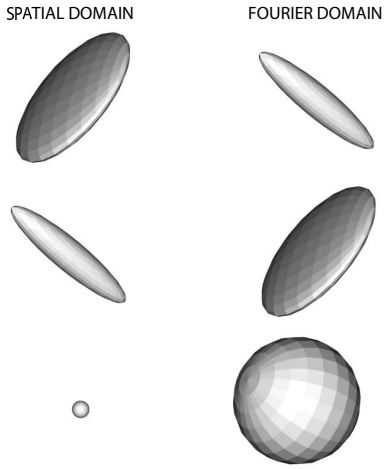


Figure 5.6: Spatial representation and Fourier energy distribution of local neighborhoods. From top to bottom: plane, line and isotropic case

5.5 Flow estimation in time sequences

Image sequences can be represented as three-dimensional spaces; two spatial and one temporal dimension. For such time sequences, a 3D plane means a line in the 2D image and a 3D line means a point in the image. As demonstrated in section 5.4, the eigenvalue-distribution of the 3D estimated orientation tensor can be used to classify the local neighborhood into a plane, line or an isotropic case. In the line case, corresponding to a moving point in the 2D image, one can project the orientation of the lines, given by $\hat{\vec{e}}_3$, onto the temporal axis, and obtain an estimate of local velocity. In the plane case only an estimate of velocity *perpendicular* to the plane, or moving line in the 2D image, can be determined. The velocity component along the moving line cannot be determined since motion in that direction induces no change in the local signal. This is commonly referred to as the *'aperture problem'*[14].

5.5.1 Extracting normal and true flow

By examining the relation between the eigenvalues, it is possible to estimate to which of the above categories the neighborhood belongs. Depending on the category, either a *true*, or a *normal* flow can be estimated. As suggested in [14], the following functions can be seen as probabilities for each case:

$$\begin{aligned} P_{plane} &= \frac{\lambda_1 - \lambda_2}{\lambda_1} \\ P_{line} &= \frac{\lambda_2 - \lambda_3}{\lambda_1} \\ P_{isotropic} &= \frac{\lambda_3}{\lambda_1} \end{aligned}$$

If the function giving the highest output is P_{plane}, i.e., we are in the moving line case, the *normal* flow is calculated by:

$$\vec{v}_{norm} = -x_3(x_1\vec{\hat{\xi}}_1 + x_2\vec{\hat{\xi}}_2)/(x_1^2 + x_2^2)$$

where

$$x_1 = \vec{\hat{e}}_1 \cdot \vec{\hat{\xi}}_1 \qquad x_2 = \vec{\hat{e}}_1 \cdot \vec{\hat{\xi}}_2 \qquad x_3 = \vec{\hat{e}}_1 \cdot \vec{\hat{t}}$$

in which $\vec{\hat{\xi}}_1$, $\vec{\hat{\xi}}_2$ and $\vec{\hat{t}}$ are the orthonormal vectors defining the image plane and time direction, respectively. If P_{line} gives the highest output, i.e. we are in the moving point case, the *true* flow is calculated by:

$$\vec{v} = (x_1\vec{\hat{\xi}}_1 + x_2\vec{\hat{\xi}}_2)/x_3$$

where

$$x_1 = \vec{\hat{e}}_3 \cdot \vec{\hat{\xi}}_1 \qquad x_2 = \vec{\hat{e}}_3 \cdot \vec{\hat{\xi}}_2 \qquad x_3 = \vec{\hat{e}}_3 \cdot \vec{\hat{t}}$$

If, on the other hand, $P_{isotropic}$ is of highest value, the neighborhood is deemed isotropic, and thus no velocity can be determined.

5.5.2 Averaging

Physical data always contain a fair amount of noise, and one way of obtaining a higher degree of certainty, is by averaging. By assuming that the image flow is not changing considerably from pixel to pixel, the orientation tensors can be averaged. This can be accomplished by component-by-component convolution using a low-pass filter function, a, i.e.,

$$\mathbf{T}_a(\vec{\xi}) = \sum_{\vec{x}} a(\vec{x}) \cdot \mathbf{T}(\vec{\xi} - \vec{x})$$

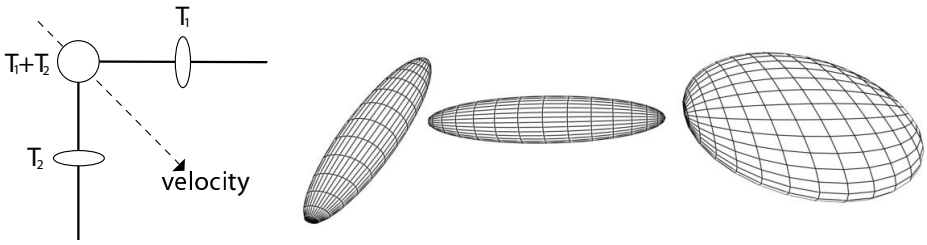
or written more compactly as

$$\mathbf{T}_a = a * \mathbf{T}$$

Since normal flow can be estimated more robustly than true flow, one can use the P_{plane} functions as certainties, thereby giving more weight to the more robustly estimated simple neighborhoods [55]. Generally, averaging two simple-neighborhood tensors, yields the result of a tensor having two eigenvectors spanning a plane, i.e., the *line case*, as illustrated in figure 5.7. The true flow can then be estimated using the resultant tensor. Using *normalized convolution* [55, 14] the weighted averaging can be written

$$\mathbf{T}_a = \frac{a * P_{plane}\mathbf{T}}{a * P_{plane}} \tag{5.8}$$

It is crucial that the averaging is done in the tensor representation. Smoothing normal *vector* velocities, will generally not prevail. As an example, the true motion in figure 5.8 would not be obtained by averaging the normal vectors because the box is an elongated square and has more vectors pointing upwards than downwards.



(a) (b)

Figure 5.7: a) Tensor addition in a corner of a moving structure. b) The same tensors shown in three dimensions.

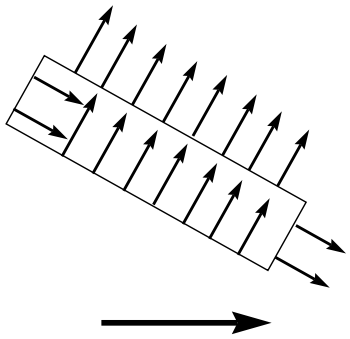


Figure 5.8: Normal flow

5.6 Adaptive filtering

The orientation tensors can be the basis for an efficient adaptive filtering approach in which the adaptive filters are synthesized as a tensor-controlled weighted summation of shift-invariant filters.

5.6.1 Tensor-controlled filters

The general tensor-controlled adaptive filter can be modeled as [14]:

$$F(\vec{u}, \mathbf{C}) = F_{lp}(\rho) + \Big[F_{ap}(\rho) - F_{lp}(\rho)\Big] \sum_{n=1}^{N} \gamma_n (\hat{\vec{e}}_n \cdot \hat{\vec{u}})^2 \tag{5.9}$$

where F_{lp} and F_{ap} are position invariant low-pass and all-pass filters, and γ_n and $\hat{\vec{e}}_n$ are the eigenvalues and eigenvectors of the control tensor, $\mathbf{C}$. If the norm of the tensor is large, the filter will reflect the shape of the control tensor, and hence retain high-pass contents along the direction of the largest eigenvalues while it will do a low-pass filtering along the orientation of the smaller eigenvalues. Small tensor norms yield isotropic low-pass filters. Figure 5.9 shows examples of resulting two-dimensional filters.

5.6.2 Tensor mapping

The control tensor, $\mathbf{C}$, in equation (5.9) can be based upon the orientation tensor by remapping its eigenvalues. A simple example would be to merely rescale the entire orientation tensor so as to fit in the filter model, e.g.,

$$\mathbf{C} = \frac{\gamma(|\mathbf{T}|)}{\lambda_1}\mathbf{T}, \qquad 0 < \gamma(\cdot) < 1 \tag{5.10}$$

The γ-function determines the overall high-pass content of the adaptive filter and is often set close to unity for neighborhoods with large signal-to-noise ratio, while dropping towards zero in noisy isotropic areas. A more lengthy discussion on the subject of eigenvalue mapping can be found in [14].

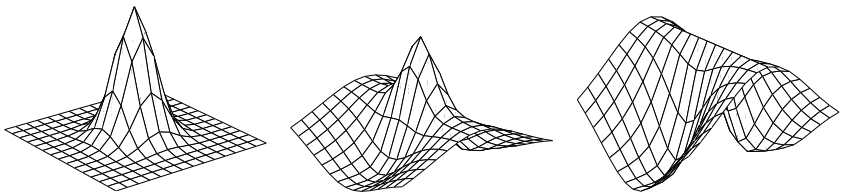
5.6.3 Adaptive filter synthesis

By using a set of fixed, spherically separable, high-pass filters,

$$F_k(\hat{\vec{u}}) = \big[F_{ap}(\rho) - F_{lp}(\rho)\big](\hat{\vec{n}}_k \cdot \hat{\vec{u}})^2,$$

directed along the same orientations as the quadrature filters, the flow-adaptive filter can be written

$$F(\vec{u}, \mathbf{C}) = F_{lp}(\rho) + \sum_k c_k F_k(\hat{\vec{u}}) \tag{5.11}$$



(a) $\gamma_1 = 0, \gamma_2 = 0$ (b) $\gamma = 0.5, \gamma_2 = 0$ (c) $\gamma = 1, \gamma_2 = 0$

Figure 5.9: Fourier-space representation of two-dimensional adaptive filters resulting from tensors describing line-like neighborhoods with different control-tensor eigenvalues

in which $c_k = \mathbf{C} \cdot \mathbf{M}_k$. Merely simple algebraic manipulations are needed to get from (5.11) to (5.9).

There is no need to actually produce the adaptive filters since the process is linear, and hence the output of the synthesized filter, f, is obtainable by weighting the outputs from the fixed filters:

$$f = f_{lp} + \sum_k c_k f_k \tag{5.12}$$

5.7 Kernel optimization

This section demonstrates how to obtain the actual quadrature filter pairs.

5.7.1 The reference function

As mentioned in section 5.1 on page 37 the ideal band-pass filter will, in the Fourier domain, be real and have zero negative frequencies. As suggested in [14] the ideal reference functions used here are of the form:

$$F_k(\vec{u}) = \begin{cases} e^{-\frac{4}{B^2 \ln 2} \ln^2(\rho/\rho_i)} (\hat{\vec{u}} \cdot \hat{\vec{n}}_k)^2 & \text{if } \vec{u} \cdot \hat{\vec{n}}_k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

where $\rho = |\vec{u}|$, $\hat{\vec{n}}_k$ is the filter directing vector, B is the relative bandwidth and ρ_i is the center frequency. A visualization of the reference function in two dimensions, with $B = 2$ and $\rho_i = \pi \cdot 2^{-1.5}$, is given in figure 5.2 on page 40. The smoothness of the function ensures spatio-temporal locality.

5.7.2 Distance measure

The kernel coefficients are optimized so as to minimize a weighted mean square distance to the ideal filter reference function, i.e., minimizing

$$\epsilon^2 = |\mathbf{W}^{\frac{1}{2}}(\vec{F} - \mathbf{G}\vec{f})|^2 = (\vec{F} - \mathbf{G}\vec{f})^t \mathbf{W} (\vec{F} - \mathbf{G}\vec{f}) \tag{5.14}$$

where $\mathbf{W}$ is a diagonal weight-matrix, $\vec{F}$ is the sampled ideal reference function, $\mathbf{G}$ is the Fourier basis function matrix, and $\vec{f}$ is a vector containing the spatial kernel coefficients. Sampling the ideal Fourier space using a density 2-3 times as high, in each dimension, as the size of the spatial filter, is in practice adequate. Further increasing the sampling density will give an insignificant effect on the resulting filter.

This is a classical over-determined inverse problem which can be solved by simply computing the partial derivatives of (5.14) with respect to the kernel coefficients, $\vec{f}$, and solving

$$\frac{\partial \epsilon}{\partial \vec{f}} = 0$$

The solution with respect to the kernel coefficients, $\vec{f}$, can, with a little time and effort, be shown to be [35]:

$$\vec{f} = [\mathbf{G}^t \mathbf{W} \mathbf{G}]^{-1} \mathbf{G}^t \mathbf{W} \vec{F}$$

5.7.2.1 Multiple space optimization

The error measure, ϵ, might incorporate multiple representation spaces. Let $\mathbf{G_n}$, F_n and $\mathbf{W_n}$ be the basis function matrix, ideal sampled reference function and weight function, respectively, corresponding to representation space n. The total error measure then becomes

$$\epsilon^2 = \sum_{n=0}^{N} (\vec{F_n} - \mathbf{G_n}\vec{f})^t \mathbf{W_n} (\vec{F_n} - \mathbf{G_n}\vec{f}) \tag{5.15}$$

which, minimized with respect to the kernel coefficients, $\vec{f}$, results in [28]:

$$\vec{f} = [\sum_{n=0}^{N} \mathbf{G_n}^t \mathbf{W_n} \mathbf{G_n}]^{-1} \sum_{n=0}^{n} \mathbf{G_n}^t \mathbf{W_n} \vec{F_n}$$

By including the spatio-temporal space, setting the corresponding reference function to zero, and using a weighting function of the form; $w(\vec{x}) = |\vec{x}|^\gamma, \gamma > 0$, one can introduce a distance measure favoring spatio-temporally localized kernels.

5.7.3 The weighting function

The Fourier-space weighting function determines the importance of a close fit for the different spatial frequencies. Generally, one ought to incorporate as much a-priori information available on the specific problem at hand in determining the weight function. Here, we will use our knowledge of the expected radial spectra, S, of synthetic images comprising of a large number of random edges or lines [28];

$$S(\rho) \propto \begin{cases} \rho^{-0.5} & \text{random lines} \\ \rho^{-1.5} & \text{random edges} \end{cases}$$

Assuming that noise presents itself in broadband terms, an appropriate weighting functions might be:

$$w(\vec{u}) = \rho^{-1} + n \tag{5.16}$$

where n relates to the expected level of noise. A more general weighting function might take into account possible band-limiting of the signal [28]. Note that $\lim_{\rho \to 0} = \infty$, ensuring that the DC-component is given top priority. Here, and for many other instances, this is of crucial importance.

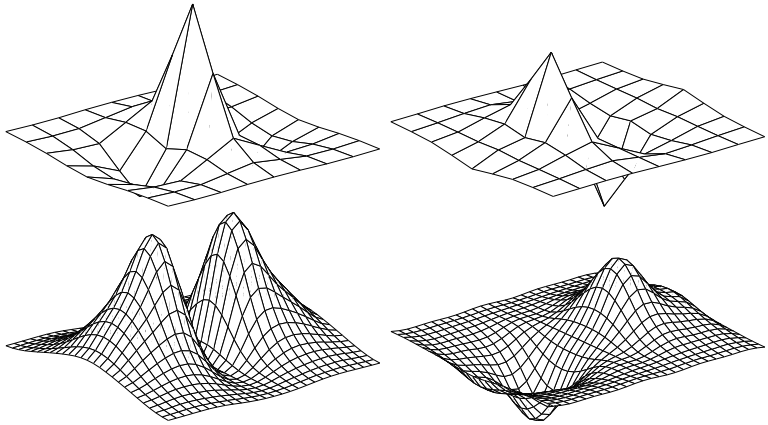


Figure 5.10: Optimized 9×9 filter kernels and their Fourier transforms

5.7.4 The resulting filter pairs

The resulting complex filter kernel can be separated into one symmetric real part, and one anti-symmetric imaginary part. Letting the output of each of these filters be q_r and q_i for the real and imaginary part, respectively, we can obtain an estimate of local orientational energy by combining the output, i.e.,

$$q^2 = q_r^2 + q_i^2.$$

The real and imaginary part of a two-dimensional 9×9 filter kernel, and their respective Fourier transforms are given in figure 5.10. The Fourier transform of the real symmetric part is real, and the Fourier transform of the anti-symmetric imaginary part is imaginary. The two parts can be thought of as generalized line and edge detectors, respectively.

Chapter 6

The General Flow-Adaptive Filter

The general flow-adaptive filter utilizes the low spatial variance of flow fields to simplify and improve filter-adaptability to spatial movement in time sequences. Before discussing the details of the filter, the more traditional ways of implementing filters in such sequences are summarized.

6.1 Spatio-temporal filtering

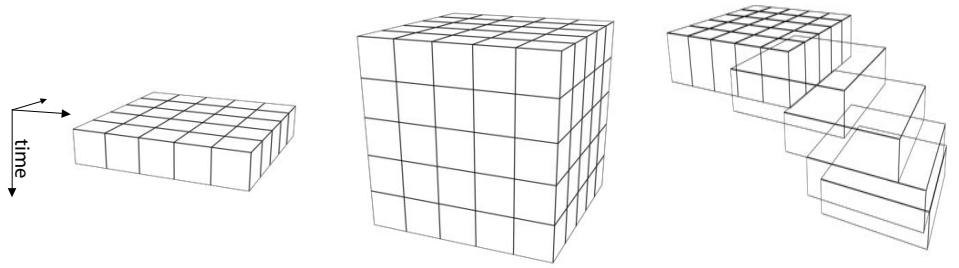
When each frame is treated individually, the general filter can be seen as a two-dimensional lattice shifted over the data, and at each point the element corresponding to the central coordinate of the lattice is replaced with a new value based entirely on the values inside the filter. The lattice, as illustrated in figure 6.1(a), then corresponds to the *realm* of the filter. In image sequences, the data to be filtered is three dimensional, two spatial and one temporal dimension, hence the filter can be extended to a cubic lattice as illustrated in figure 6.1(b).

6.1.1 Simple adaptive 3D filtering

As mentioned in section 3.4, simple ways of adapting the filtering process to local signal characteristics are to use varying filter sizes or linear combinations of multiple filters. Such schemes usually operate on simple isotropic models, like whether the filter is in a high-variance, or an homogeneous, neighborhood. Neighborhoods of high variance are often filtered by either a size-reduced or an edge preserving filter.

6.1.2 Anisotropic window adaption

By weighting the elements of the lattice, the window itself can be made adaptable to different signal structures, i.e., it can be aligned to edges and three-dimensional



(a) Two-dimensional filter (b) Three-dimensional filter (c) General flow-adaptive filter

Figure 6.1: Two- and three-dimensional filters

structures as discussed in section 3.4.3. Schemes like these often assume linear structures in the three dimensional data sets, or have complex underlying models requiring relatively large amount of data, that is, large lattice sizes, to ensure some stability.

6.2 General flow-adaptive filter

A novel filtering technique based on estimated flow fields will now be presented. First, the reliability of the flow fields must be assured.

6.2.1 Flow-field assumption

One important assumption, which is valid for most image sequences and is analog to the assumption mentioned in the introduction to chapter 5, is that the spatial variance of the local flow is relatively small. That is, the local flow rarely changes much from pixel to pixel. This assumption renders it likely that our *estimate* of the local flow can be made reassuringly accurate, since both the use of larger, i.e., more robust, windows in the flow-field estimation and spatial filtering of the resulting flow fields can be implemented. Enough a-priori information about the flow fields can make the above assumption superfluous.

The filtering process described below does not rely on any single technique for the estimation of the flow fields, and both block-matching and energy-based schemes have been used in this thesis. The important point here is just that it is available, and that it has a certain reliability.

6.2.2 Filtering principle

The general principle is to spatially adapt the entire filter lattice to possibly complex spatial movements in the temporal domain by incorporating local flow-field

estimates.

The filter lattice is placed in the data sequence according to the following procedure:

1. Place the middle, spatial *slice*, or plane, centrally upon the output pixel.
2. From the central pixel, let the flow field guide a spatial movement, go one temporal step down, and place the next filter plane centrally at that location. Now, let the local flow at that last spatio-temporal position, guide another spatial move, go one more temporal step, and place the next filter plane centrally at that location. Continue until all the planes from the middle and downwards have been placed.
3. The planes from the upper half of the filter are placed in the same manner, except that now the temporal direction is negative. If separate flow fields for the temporal directions are not available, the orientations of the flow field are merely reversed.

An example of a resulting lattice is shown in figure 6.1(c). Note that it is referred to *local* flow, so that the estimates change at different spatial coordinates.

The technique just presented is independent of the type of filtering deployed upon the resulting lattice, e.g., whether it is an average, median or a more complex-modeled filter. In a sense, the general flow-adaptive filter *transforms* the image sequence into a more manageable motionless state.

An example is illustrated in figure 6.2. The $3 \times 3 \times 3$ filter lattice is shown frame by frame. The first row shows which pixels the original cubic lattice encapsulates. The lower row shows which pixels are selected when utilizing an estimate of the local flow and the novel filtering procedure. The pool of relevant pixels are much larger in the novel approach, yielding a more correct filter output.

6.2.3 Implications

Some of the direct implications of the proposed filtering scheme are that it:

- Overcomes the limitation of the often-made assumption of linear structures
- Gives a higher temporal stability than the complex-modeled filters
- Allows simpler within-filter models to trace complex temporal movement
- Allows smaller filter sizes since the lattice is filled with more stable pixels
- Known filters can be used directly on the resulting filter data

Of course, the computational costs needed to estimate the flow fields cannot be disregarded, but there are several highly efficient estimation techniques, including [51] for correlation-based estimation and [24] for the energy-based approach.

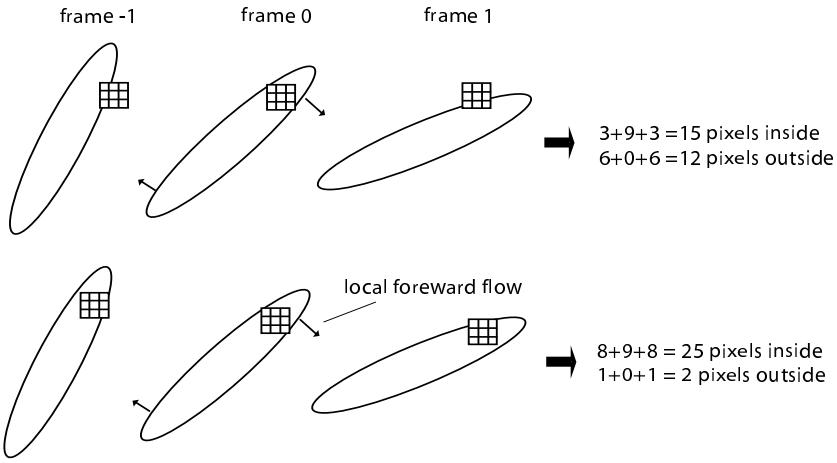


Figure 6.2: Example of how the flow-field estimates improve the local, spatial stability of the pixels encapsulated by a $3 \times 3 \times 3$ filter

6.2.4 Example: Discontinuity filter

In the case of the strictly temporal discontinuity filter discussed in section 4.1, the flow-adoptive approach can make a noticeable difference. As illustrated in figure 6.3, objects in ultrasound image sequences are likely to move, making the pool of samples collected in the temporal direction contain values from both inside the object, the border of the object, outside the object, and even from other objects. By adopting the flow-adaptive filter and hence letting an estimate of local flow guide an additional spatial movement at each frame, this error can be reduced. The pool then consists of intensity values that spatially are more locally stable within the moving objects.

The principle of letting the local flow-estimate guide the spatial movement when selecting the set of values used in the filtering process is illustrated in figure 6.4. At frame zero, for instance, the local flow indicates that the object moves slightly to the right, and therefore at frame one the sample is selected correspondingly in that direction. This gives a more locally stable path, yielding more correct filter outputs.

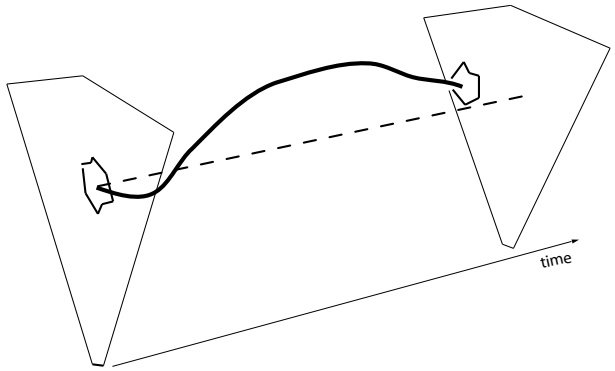


Figure 6.3: Illustration of an object's true path (solid line) and the path of which the samples are selected in a strictly temporal fashion in an ultrasound image sequence

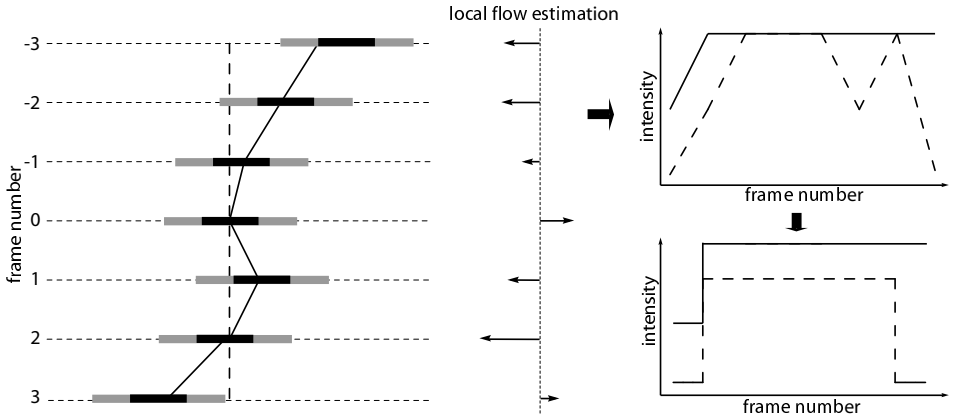


Figure 6.4: How the flow-adaptive approach can render more correct results in the case of discontinuity filtering. The dashed line indicates a path of straight temporal orientation, and the full line a flow-estimate guided path. The right side shows the resulting intensity values together with a piecewise constant reconstruction.

Chapter 7

Experimental Materials and Methods

This chapter describes the experiments conducted in this thesis. The objective was to see whether the flow-adaptive filter approach of chapter 6 improves filtering performance. Both the filtering of real ultrasound-image sequences and synthetic sets were carried out. Synthetic sets open for a larger scope of filters to be tested for improvement using the flow-adaptive approach and make it possible to generalize the results to a non-ultrasound context.

The filtering of the real ultrasound sequences was focused on the discontinuity filter described in section 4.1, both in its original form and using the novel flow-adaptive filter method. The discontinuity filter was elected based on its excellent performance in temporal filtering of ultrasound sequences.

Three simple non-adaptive filters, the average, Gaussian-shaped convolution and the median, were implemented using the flow-adaptive technique and tested on the synthetic data sets. These filters were picked based on their simplicity, and their lack of underlying models and parameters. Hence, indications of improvement using flow-adaptive principles are not cluttered by discussion of parameter settings, and their implementation is straightforward. In addition, the tensor-based adaptive filter described in section 5.6 was tested using the flow-adaptive approach. This filter was chosen due to its close relationship with the energy-based flow-estimation technique, and to show that even fairly advanced filters improve with the flow-adaptive procedure. Testing how the general flow-adaptive filtering technique improves the entire spectrum of filters is not the intention of this thesis.

The first section describes the acquisition of the all-important flow fields.

7.1 Estimating the flow field

The flow fields have been estimated both by using speckle tracking and the energy-based technique discussed in chapter 5. Details of their implementation are discussed next.

7.1.1 Energy-based estimation

The energy-based method of estimating flow first probes the local three dimensional data set for gradients, resulting in a tensor representation of the local neighborhood. The tensors are then filtered before local flows are extracted.

7.1.1.1 Quadrature filters

The quadrature functions were obtained by minimizing a weighted square distance to the reference function (5.13), using the weights given in (5.16), where the latter was chosen to maintain high generality. The sample resolution was twice that of the filter size.

The optimized filters had a size of $7 \times 7 \times 7$. This gives an acceptable computational load, allows robust estimation, and corresponds well with the size of the structures in the data sets.

7.1.1.2 Tensor filtering

An alternative to the certainty weights used in the normalized convolution (5.8), P_{plane}, is [55]:

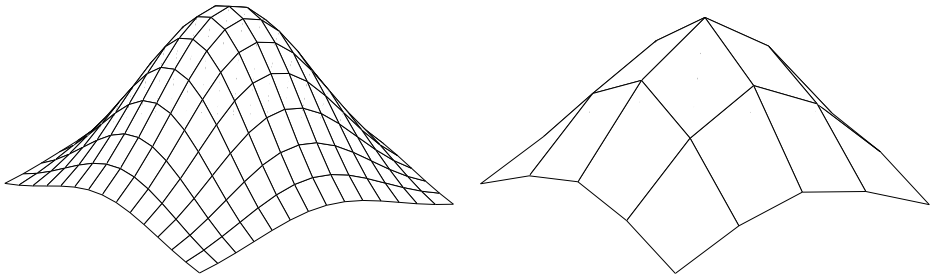
$$c = \frac{\lambda_1 - \alpha\lambda_2}{\lambda_1}$$

where the parameter $\alpha > 1$ allows favoring tensors describing the more robustly estimated plane-like neighborhoods.

The tensor filtering was performed using a 15×15 Gaussian function as shown in figure 7.1(a), and using $\alpha = 1.4$.

7.1.1.3 From tensors to flow

Before extracting the flow, a small constant, ϵI, was added to the tensors as proposed in [55]. This constant can be seen as a regularization term. Small tensors will be deemed isotropic since they will become nearly 'round', and consequently be dismissed. The experiments were done with an ϵ equal to about one percent of the largest tensor norm in the data sequence. The flow was extracted from the tensor as described in section 5.5.1.



(a) 15×15 kernel used in tensor filtering (b) 5×5 kernel used for flow filtering

Figure 7.1: Two-dimensional filtering functions

7.1.1.4 Flow field smoothing

To ensure a more stable and correct flow field, filtering by normalized convolution was performed. The application function is shown in figure 7.1(b), and the certainty of each flow estimate was set to

$$c = |\mathbf{T}| \cdot (P_{line} + \delta)$$

where $|\mathbf{T}|$ is the norm of the tensor and δ is a small constant preserving strictly normal-flow neighborhoods. Thus, the certainty value favors large tensors describing line-like neighborhoods.

Other popular ways of filtering flow fields are based on *vector medians* [1]. A simple implementation of a *vector directional filter*[42] was tried, but abandoned, since the difference in performance was negligible for the type of experiments conducted in this thesis.

7.1.2 Speckle tracking

The flow fields for the real ultrasound sequences described in section 7.3.2 were also estimated using the speckle tracking technique explained in section 4.2. The kernels had a size of 15 pixels radially and 7 pixels laterally, and the search distance was 10 pixels in both directions. The normalized correlation was used for the best-match criterion. The resulting flow field was filtered in the same manner and with the same convolution function as in the energy-based case, but by using the peak correlation coefficients as weights in the normalized convolution. Flow in both positive and negative temporal direction was extracted.

The kernels were set large enough to ensure relatively robust flow estimates, while still allowing a certain spatial variance. Optimal speckle-tracking settings for every sequence were not pursued. The focus was on simplicity, and showing that even rough motion estimates are adequate for the prevalence of the flow-adaptive filter.

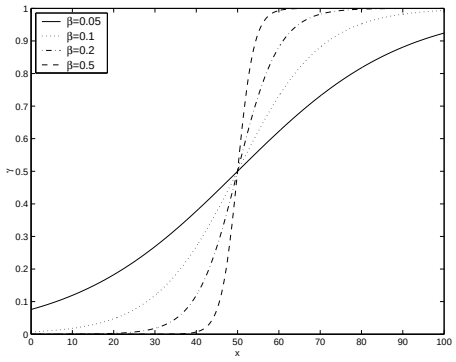


Figure 7.2: The gamma scaling function used in obtaining the control tensor with $\alpha = 50$ and varying β

7.2 Filter parameters

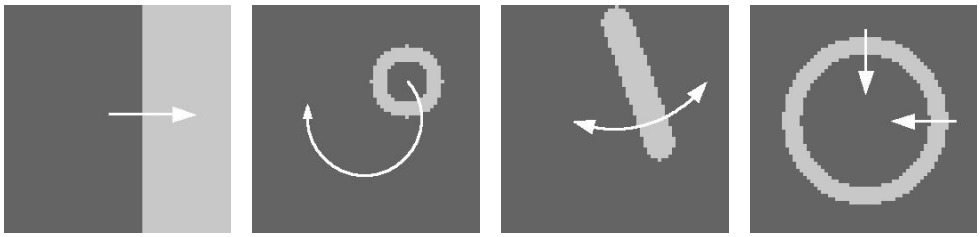
7.2.1 Discontinuity filter

The size of the time window used in the filtering algorithm described in section 4.1 was set to 7 samples, and placed in a cyclic manner. Constant reconstruction was performed with 2 discontinuities, as in figure 4.2. The *leakage* was set to 0.3, that is, 30% of each of the boundary pixels were included in the final filter output.

The length of the filter was kept short to justify the use of only two discontinuities, reduce computational complexity, and assure good relative performance of the straight temporal filter. The leakage was set low enough to not introduce mentionable motion blur, but high enough to make an impact on the filtering. It is shown that, even with these settings giving fairly optimal results using the original filter, the flow-adaptive technique prevails.

7.2.2 Non-adaptive filters

The simple non-adaptive filters, the average, Gaussian shaped convolution and the median, all had a spatial size of 3×3 and 5×5, with a varying temporal extent. The filter sizes were kept small to fit the image structures. Note that the quadrature filters of section 7.1.1.1 have a different function and can hence be of a larger size.



(a) (b) (c) (d)

Figure 7.3: Still frames from the synthetic data sets. Direction of motion is indicated by arrows.

7.2.3 Tensor-based filter

The implemented filter based its adaptive filter synthesis on the same orientation tensors used in the energy-based flow estimation. The low-pass filter of equation (5.9) was Gaussian shaped while the all-pass filter was a Dirac-delta pulse. The high-pass filters of equation (5.12) were obtained by minimizing the squared distance to a sampled reference function with twice the resolution. The size of the high-pass and low-pass filters were all $7 \times 7 \times 7$. The control tensor was derived from the orientation tensor by the scaling function of equation (5.10), in which

$$\gamma(x) = \frac{1}{1 + e^{-(x-\alpha)\beta}} \tag{7.1}$$

where α reflects the noise, and β controls the transition period from low-pass to high-pass filtering. Figure 7.2 shows the scaling function with varying β. The experiments were done with $\beta = 0.05$.

The adaptive tensor-based filter has to be of a larger size than the non-adaptive filters of section 7.2.2 because the high-pass filters require a certain amount of coefficients to resemble their reference functions in an adequate way. As a consequence, the synthetic sequences used for testing the tensor-based filter had larger image structures matching the increase in filter size.

7.3 Data sets

7.3.1 Synthetic sets

In addition to the real ultrasound data sets described in section 7.3.2, synthetic image sequences were generated. This provided a *sterile* environment with a complete control of all parameters, and a direct error metric as discussed in section 7.4. To assure reliability, all experiments using the synthetic sets had at least 20 iterations.

7.3.1.1 The sequences

Four classes of sequences, each consisting of 28 frames of size 64×64 pixels, were generated:

Moving wall Figure 7.3(a) shows one of the frames in a simple sequence consisting of a white, vertical wall moving from left to right. The only adjustable parameter is the velocity of the moving wall. If otherwise not explicitly stated, the velocity parameter is 2 pixels per frame for this sequence, that is, the *standard parameter* is a velocity of value 2.

Circulating annulus Figure 7.3(b) depicts a frame from a sequence in which a white annulus moves around the image center in a circular fashion. The parameters are the radius of movement, the annulus size, given by inner and outer radius, and the angle increment per frame of the movement. The *standard parameters* for this sequence are a movement radius of 16 pixels, an inner radius of 6, outer radius of 10, and a $\frac{\pi}{16}$ per-frame angle-increment giving about a 3.1 pixel movement of the annulus center per frame.

Pendulum Figure 7.3(c) displays a still frame of a moving pendulum sequence. The parameters are the pendulum width, the drop angle, and a movement step per frame to regulate velocity in the sequences. The *standard parameters* are a width of 9 pixels, $\frac{\pi}{6}$ drop angle, and a movement per step making the sequence repeat itself every 24th frame.

Pulsating ellipse Figure 7.3(d) is an image from a sequence where a white ellipse is changing size in a pulsating way. The parameters are the distance between the foci, the size of the inner and outer major axis, maximum increase in size along the major axis, and a step value to regulate inter-frame movement. The *standard parameters* are, in the same order as above, 8, 12, 22, 25 pixels and a movement step making the sequence repeat itself every 16th frame.

In addition, '*ball*' sequences consisting of disc-shaped structures moving from left to right were generated when the effect of structure sizes was investigated. The parameters were the diameter and velocity of the ball. The frames for these sequences were elongated when high-velocity environments were generated.

The first sequence class, the moving wall, provides an environment for studying filter behavior on sequences consisting of purely simple unidirectional flow, while the other classes provide sequences with multifarious non-simple movements. All classes, except the first, allows the size of the image structures to be manipulated.

The sequences were kept with floating point precision at all times. That which

appears grey in the images of figure 7.3 has a value of 50, and what appears white has a value of 100.

Image smoothness The borders of the image structures in the sequences above are step-wise, that is, goes from background to foreground intensity in one pixel. To see whether this had any effects on the results, sets of sequences where the edges had been smoothed were included in the experiments. The smoothing was done by filtering the sequences with a spatial 3×3 average filter.

7.3.1.2 Noise models

Three different noise models were used; Gaussian additive, Gaussian-based multiplicative, and impulse noise. More precisely,

$$x = s + n, \qquad x = s + \sqrt{s} \cdot n \tag{7.2}$$

are the additive and multiplicative models, respectively, where s is the uncorrupted signal and n is an independent, zero-mean, Gaussian random variable, i.e., its probability density function is given by

$$f_n = \frac{1}{\sigma\sqrt{2\pi}}\, e^{\frac{-n^2}{2\sigma^2}}. \tag{7.3}$$

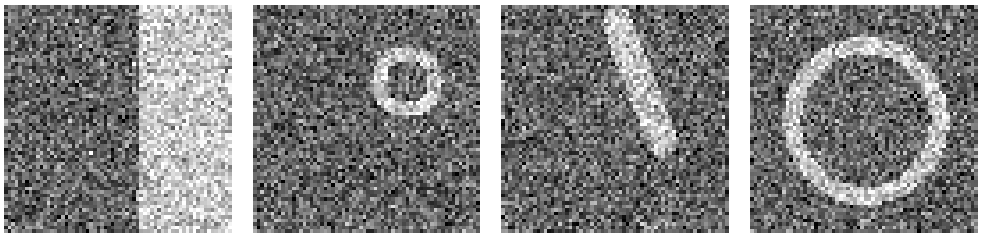
The multiplicative noise model in (7.2) can be used to mimic the noise characteristics of ultrasound images [29]. The impulse noise model is given by

$$x = \begin{cases} d & \text{with probability } p \\ s & \text{with probability } (1-p) \end{cases}$$

where p is the probability that a distortion, $d \neq s$, will occur at each sample. The d-value was set either to 25 or 125 with equal probability.

Unless explicitly stated, the signal-to-noise ratio (SNR) was set to 8dB for the additive and multiplicative noise, that is, $\sigma_{add} = 20$ and $\sigma_{mult} = 2$ in the density function (7.3). The impulse-noise model had $p = 0.1$ as its standard parameter. Figure 7.4 shows the synthetic sets corrupted by additive Gaussian noise with SNR=8dB.

Other methods producing even more speckle-like noise were omitted. The required models for such an approach would have to assume a lot of underlying parameters, and since the focus here is on the images themselves, they would not make much difference anyway. However, if the error metric was based on the corrupted images instead of the approach in section 7.4.1, e.g., some kind of texture-classification scheme, it would be critical to have an accurate noise model.



(a) (b) (c) (d)

Figure 7.4: The images in figure 7.3 corrupted by additive white Gaussian noise with SNR=8dB

7.3.2 Ultrasound sequences

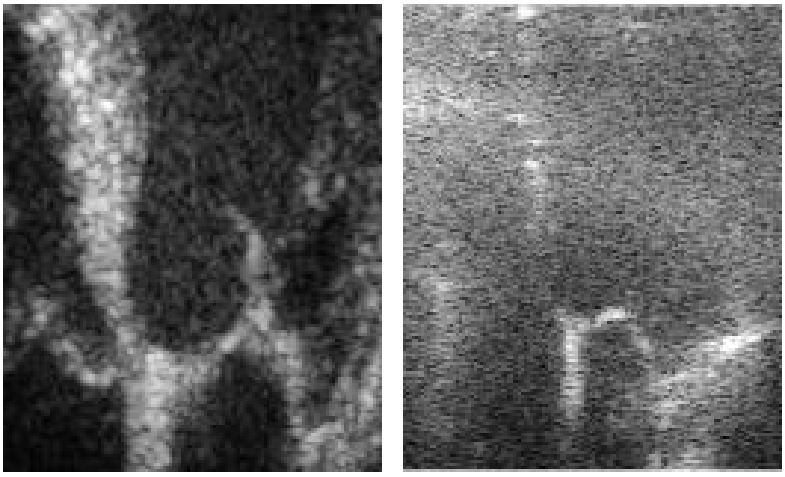
The clinical ultrasound image-sequences that were filtered upon were of beating hearts taken by a 64-element probe with a central frequency of 2.5MHz. The end-depths, the number of beams used for each image frame, as well as the frame rate differed between the sequences. Typically the non-scan-converted images had a size of about 300×150 to 600×200 pixels, and the number of frames in a sequence covering one heart cycle was between 28 and 50 images. The experiments were performed directly upon the non-scan-converted images without producing the geometrically correct outputs. This was done to avoid any unnecessary tampering of the data before the experimental filtering was conducted. The benefits of doing this over-shadowed the geometrical distortion disadvantages when performing the speckle tracking.

Note that the filtering procedures investigated in this thesis all operate exclusively in the image domain, making the technical details of the ultrasound-scanner settings less important. Of course, the frame rate, for instance, is important, but only indirectly. An increased frame rate reduces the inter-frame structural movement, and that is what is important for the filters applied here, not the frame rate itself.

Varying clinical conditions of the patients yielded the necessary variance in the visual quality of the images needed for the error metric described in section 7.4. This error metric requires the sequences to be grouped into classes of either high or low visual quality. Examples of still images from both categories are found in figure 7.5.

7.4 Error metrics

How the filters perform are measured differently in the synthetic and the ultrasound sequences.



(a) High quality (b) Low quality

Figure 7.5: Excerpts of four-chamber-view images of the cardiac function from the high- and low-quality classes. The image in b) is severely degraded by reverberation. Note that the images are non-scan-converted.

7.4.1 Synthetic sets

The success of the filters on the synthetic sets were measured by the mean-square error between the filter output and the original uncorrupted, or noiseless, sequence normalized by the noise level, i.e.,

$$\text{nmse} = \frac{\sum_{x_f \in S} (x_f - s)^2}{\sum_{x \in S} (x - s)^2} \tag{7.4}$$

where x_f is the filtered pixel taken over a set of pixels, S, s is the original uncorrupted signal, and x is the noisy data that has been used as filter input.

S was set to contain all pixels in the noiseless sequence which had the property that if a square, spatial window of size 3×3 was placed centrally upon it, there would be variance within the window pixels. This makes the effective metric regions contain only the edges, that is, the pixels before and after any transitions from background to foreground and vice versa. The non-adaptive filters were also examined with a variance-detection filter of size 9×9. Such a size exposes possible motion blurring, while still assures that large homogeneous areas, e.g., outside the objects, which is not of interest here, do not dampen the error metric.

In addition, the filtered results were inspected visually. Other application-dependent error metrics could have been implemented, but the above methods, when the ideal,

noiseless sequences are available, are quite general and robust. Implementing a wide range of all possible error metrics is not within the scope of this thesis.

7.4.2 Ultrasound sequences

To test whether the novel flow-adaptive filtering scheme excels in the case of the real ultrasound sequences, the following two postulates were put to the test:

1. For noisy sequences the filtered regions of motion should be smoother with the novel than with the strictly temporal approach. I.e., the filtering should decrease the local variance, and produce spatially more stable output.

2. For images with little noise, the filter should retain fine structures in the original image, also for regions of motion.

To verify the first postulate in the present case, measures of local variance were put into different velocity classes. Each neighborhood had their local absolute velocity rounded to the nearest integer, and their local variance estimate was said to belong to that velocity class. This way the spatial smoothness can be displayed as a function of absolute velocity. To make sure that gradient neighborhoods were not given excessive non-smoothness estimates, the usual variance was replaced with how well the local neighborhood could be represented by a plane, that is,

$$\text{non-smoothness} = \min_{a,b,c} \sum_{x \in S} (x - (ax + by + c))^2, \quad S = neighborhood$$

Figure 7.6 shows a neighborhood and a plane minimizing the squared error criterion. The size of the local neighborhoods was set to 5×5. If computational efficiency is an issue, the coefficients in such polynomial fitting schemes can be estimated by a few simple convolutions [31, 16].

The second postulate was verified using the local squared difference between the original, nearly noiseless, sequence and the output of each of the two filtering techniques, again using different velocity classes. The size of the local neighborhoods was set to 11×11, which is big enough to detect structure but small enough to allow velocity variance. The novel scheme should reduce filtering artifacts, and thus give a smaller value in the above method than the old approach for neighborhoods with motion. Of course, such a metric does not incorporate perceptual considerations and is hence prone to detect errors like simple shifting, but it must be noted that the squared error is taken over *local* windows and not entire images, and that such spatial shifts could be argued to be qualified errors. Other ways of measuring whether fine structures are retained can be found in [50] and references therein.

As with the synthetic sets, visual judgements were also performed. Other intuitive approaches would be to let clinical professionals judge the resulting images

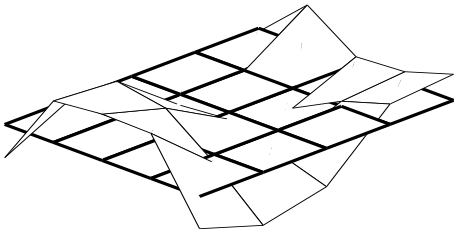


Figure 7.6: Fitting a plane to the local neighborhood using the minimum squared-error criterion

for diagnostic usefulness, or use the images as input in programs performing automatic measurements. The first approach is infeasible due to the relatively low priority of such a task among busy clinical professionals. The other approach is very technique-dependent, and doing a wide study on the subject would go beyond the limits of this thesis.

Chapter 8

Results and Discussion

This chapter presents the results and discussions of the experiments. Although they reside in the same chapter to ease the reading, the result and discussion parts are kept separate. The main focus will be on the synthetic data sets. Those sets have a more direct and robust quality metric, hold important free parameters, and they make it possible to generalize the results to non-ultrasound contexts.

The consequences of the parameters of the synthetic sequences are discussed separately, beginning with the noise models and varying SNR. Then the other parameters; velocity, structure size, image smoothness, metric region and filter parameters are discussed before a general conclusion to the visual inspection is given. It will be shown that the flow-adaptive approach substantially improves the performance of all the implemented filters.

The abbreviations *fa*, *nfa* and *nmse* are used extensively throughout the text and they are short for *flow-adaptive*, *non-flow-adaptive* and *normalized mean-square error*, respectively.

8.1 Synthetic sets

The details of the synthetic sets, including their *standard parameters*, are described in section 7.3.1. Note that unless otherwise stated, only edges are included in the metric, which accounts for the occasional above 1.0 nmse. The nmse in more homogeneous regions is of course below 1.0.

Most of the plots omit showing the results for the Gaussian-shaped convolution filter because its introduction would clutter the figures and add limited additional information. The Gaussian-shaped filter can be argued to be in the same class as the average filter, with similar overall properties except for slightly better edge-preserving capabilities due to the central weighting of the Gaussian-shaped convolution kernel.

8.1.1 Noise characteristics

The synthetic sequences were corrupted by the three different noise models of section 7.3.1.2 with varying SNR. Figure 8.1 shows the results for the circulating annulus sequence using the additive Gaussian and the pulse noise model. Note that the metric is the non-normalized mean square error, as opposed to the normalized alternative used elsewhere. The results for the circulating annulus using the multiplicative model is found in the appendix.

8.1.1.1 Results

The flow-adaptive filters had a lower nmse than their non-adaptive counterparts for all the sequences with their *standard parameters* for all the noise levels. The results using the multiplicative noise model gave almost identical results as in the additive case, with merely a slight overall reduction of the nmse. Such a result was also apparent for the smooth images discussed in section 8.1.4, and hence justifies the extensive use of one noise model in most of this chapter.

Referring to figure 8.1, it is evident that none of the filters retain the edges perfectly even in the no-noise environment. Further, the average filters have a very straight curve for both the additive and the pulse-noise models, with the fa version being quite below its nfa counterpart. The median filters have for the additive case a slight improvement going from no noise to 15dB SNR, before they both get steadily worse as the noise intensifies. In the pulse case, the gap between the fa and the nfa median widens with higher noise levels until a certain point, about $p = 0.30$, where the gap narrows. In the case of extreme noise, the discontinuity filters have a dramatic increase in nmse, and in both graphs the average filter handles such extremes the best. Generally the gaps between the fa and nfa filters decrease as the noise level becomes more severe. This is especially prominent for the pulse model.

8.1.1.2 Discussion

The overall low nmse for the fa filters indicate that the energy-based flow estimation procedure is quite robust, at least for sequences with image structures of a certain size, demonstrating the prevalence of the general fa filter in various noise environments. For the slight scaling of the nmse using multiplicative noise, note that even though the multiplicative and additive noise models have an equal *peak* SNR, the overall multiplicative noise is lower due to the nature of equation (7.2).

In the case of zero noise, the *leakage* parameter of the discontinuity filter, discussed in section 8.1.6, accounts for much of the nmse. The rest is accounted for by the rapid changes from foreground to background intensities in the temporal path due to the velocity, the small size and the complexity of the structure in this sequence. The fa version of the discontinuity filter using zero leakage is flawless in

this particular no-noise sequence. The stability of the plots for the average filters is a result of the large number of coefficients included in the average calculation and the zero-mean property of the noise distribution. It should be noted that median filters in low-noise environments are especially penalized by the employed metric, since even though the images seem visually perfect, some of the border pixels might have been eroded, adding greatly to the nmse. For the particular sequence of figure 8.1, it is in addition a source of deficiency for the median that the image structure is small and of high velocity.

The improvement for the median filters from zero to low additive noise can be explained by the zero-mean noise property, since a little noise can make the window median closer to the window average in the cases where the alternative is complete erosion. For the pulse noise, this effect is not present, since the added noise pixels all have a value further from the average than the original pixels. The gap increase between the nfa and fa median filters in the first intervals for the pulse-noise case can be explained by the fact that the fa filter window encapsulates a larger fraction of '*correct*' pixels, either foreground or background, than the more unstable nfa filter, and is hence more able to resist the noise increase. This gap increase is not seen for the average filter, since, as explained in chapter 3, the heavily-tailed pulse-noise distribution is handled better by the median filter.

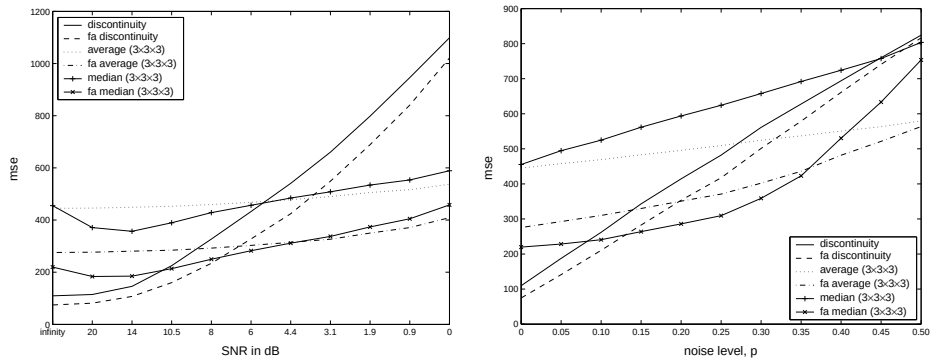
In extreme noise, the poor performance of the strictly temporal discontinuity filter is because more than one-pixel spatial extent are needed to dampen the noise. Since the pulse noise consists of equally distributed very high and very low intensities, the average filters are not as prone to error as the median, which has to *pick* one of the window intensities as output. The general equalization of the fa and nfa filter-results as the noise gets more prominent is due to the resulting loss of flow estimation, that is, the image sequences are too noisy to make exact motion estimates. Flow estimation depends furthermore on the size of the moving structures, as discussed in section 8.1.3.

8.1.2 Structure velocity

The effects of varying the velocities of the image structures in the generated sequences were investigated. Figure 8.2 shows the nmse for the synthetic sequences with varying structural velocities using the 3×3 variance-detection filter. Except for the velocity, *standard parameters* were used in the generation of the sequences.

8.1.2.1 Results

The flow-adaptive filters had, for large velocity intervals, generally a lower nmse than their non-flow-adaptive counterparts. The median outperformed the average filter most of the time. The gaps between the graphs of the fa and nfa filters were wider in the wall and annulus sequences than that of the pendulum and ellipse.



(a) Additive Gaussian noise (b) Pulse noise

Figure 8.1: Non-normalized mean squared-error metric using the 3×3 variance-detection filter with varying noise levels on the circulating annulus sequence with *standard parameters*.

Figure 8.2(a) shows that for the wall sequence there is a step-edge worsening for the nfa discontinuity filter from a velocity of zero to one pixel per frame. Similar rapid increase in nmse is found for the nfa average and median filters, but in the velocity interval from 1 to 2. After their respective nmse increments the errors stabilize. The wall results for the simple fa filters are quite stable until a velocity of 5, where they worsen before they again stabilize at 7 pixels per frame at the same level as for the nfa filters. The fa discontinuity filter has a similar behavior, except that the increase of the nmse starts at a velocity of 4.

Figure 8.2(b), which shows the results for the annulus sequence, has a more noticeable gap widening between the fa and nfa graphs as the velocity increases, until a certain point where the gap narrows. Further, the nfa filters show a nmse decrease in the angle-increment interval $\frac{\pi}{48}$ to $\frac{\pi}{24}$. A similar effect is found for the ellipse sequence in figure 8.2(d).

Figure 8.2(c) reveals a small decrease in the average-filter nmse from no motion at all to the next step along the velocity axis. The median filter displays a similar property, except that the nmse is not decreased, only stabilized in that same interval. At the highest velocity, the nmse for the fa discontinuity filter rises above its nfa counterpart.

8.1.2.2 Discussion

The prevalence of the fa filters in wide velocity intervals is apparent. The size of the intervals is determined by the capacity of the flow-estimation technique, and

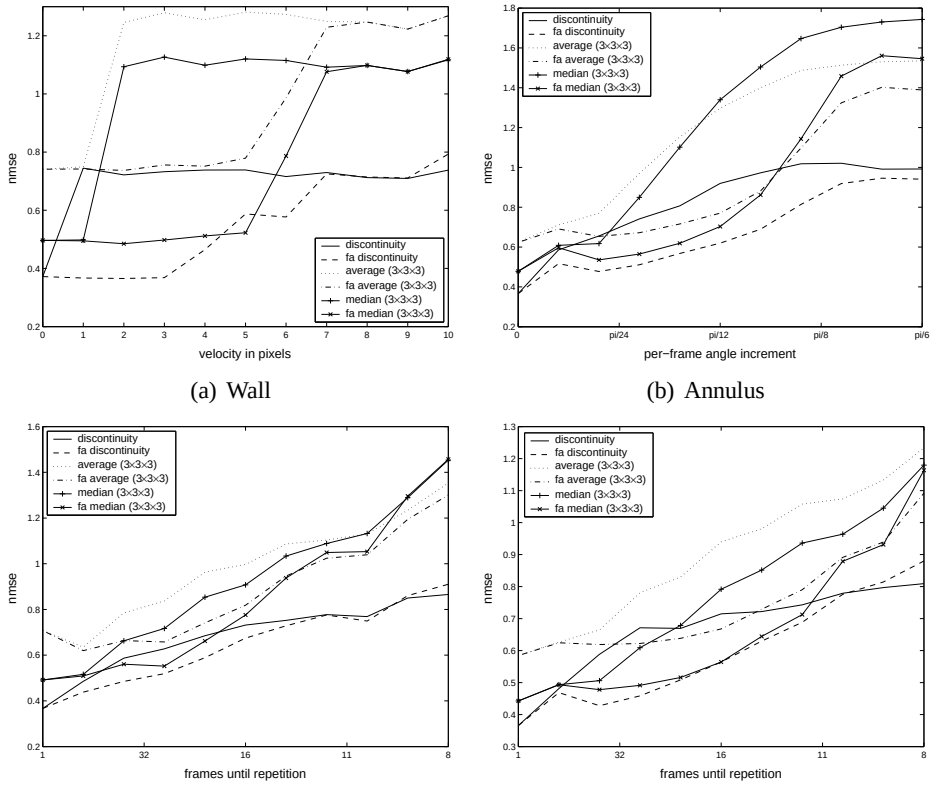
is related both to the upper absolute values of the velocity and to the inter-frame velocity consistency. The results from the wall sequence shows that the upper absolute velocity limit is about 6 pixels per frame. Recalling that the quadrature filters used in the energy-based flow estimation has a spatial extent of 7×7 pixels, the failure in detecting a 7 pixel per frame movement is obvious. Of course, using some kind of spatial coarse-to-fine strategy could overcome such upper velocity limitations [24]. The results for the pendulum and the pulsating ellipse sequences show that even though the upper absolute velocity is below the 7 pixel barrier just discussed, the swift change of direction makes the energy-based flow-estimation technique fail.

The edge-preserving properties of the median filter, as discussed in chapter 3, accounts for the low nmse of the median compared to the average filter. This is especially observable in the large-structured wall sequence. The narrower gaps between the nmse for the fa and nfa filters in the pendulum and ellipse sequences can be explained by the low velocity in many of the frames in those sequences.

The sudden increase in nmse for the nfa discontinuity filter from zero to one velocity, and the subsequent stability, for the wall sequence can be explained by the following: When the velocity is zero, the entire filter window is filled with '*correct*' pixels, i.e., all are either foreground or background pixels, yielding the low nmse result, but when the velocity is 1 or more pixels per frame, the filter windows at the edges now consist of one more '*correct*' than '*incorrect*' pixel regardless of velocity. Hence the stable subsequent error output. For the average and median filters, which both have a spatial and temporal extent of 3 pixels, figure 8.3(a) can illuminate the reason for the edge-like behavior of the nmse at velocity 2 pixels per frame and the subsequent stability. The figure illustrates how the filter window is filled with background and foreground pixels in varying velocity neighborhoods. The ratio of background and foreground pixels is the same for windows in neighborhoods of velocities zero and one pixel per frame, but for neighborhoods of velocity 2, the ratio turns more unfavorable. The proportions remain the same regardless of further velocity increments.

The reduction of fa nmse in the angle-increment interval $\frac{\pi}{48}$ to $\frac{\pi}{24}$ in figure 8.2(b) can be explained by the slightly poorer performance of the energy-based flow estimation for low velocities. In such low velocity neighborhoods the tensor norms are smaller and thus more prone to noise inaccuracies.

In figure 8.2(c), the decrease in nmse for the average filters from no to a slight velocity can be explained with the help of figure 8.3(b). The velocity is not large enough to cause serious performance degradation, but now many of the frames depicts the pendulum standing diagonally: At pixel level, it is revealed that the 3×3 variance-detection filter used in the metric will include pixels at locations similar to the one marked as a $\times$ in the figure, and the 3×3 average-filter window en-



(a) Wall (b) Annulus

(c) Pendulum (d) Ellipse

Figure 8.2: Normalized squared-error metric using the 3×3 variance detection filter on various synthetic sequences. *Standard parameters* were used except for the incrementing velocities.

capsulating that pixel will contain a large number of '*correct*' pixels yielding the reduced nmse.

8.1.3 Structure size

This section describes the results of varying the image structure sizes of the synthetic sequences. Figure 8.5 shows the results for the pendulum and ball sequences. Additional data is found in the appendix.

8.1.3.1 Results

All the sequences had a notable reduction in fa nmse as the structure size increased, until a certain point where the error stabilized. The nfa filters, especially those with more than one pixel spatial extent, did also benefit from the increase in structure size, but somewhat in a more limited way. Further, the point of stabilization for the

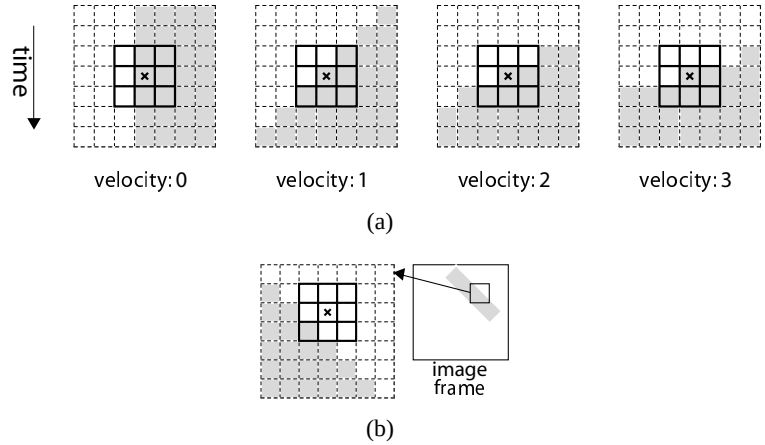


Figure 8.3: Figures aiding the explanations in section 8.1.2. a) Slices in the temporal-spatial plane illustrating the behavior of the 3×3 filters for varying velocities. b) The diagonal standing of the pendulum increases the number of pixels included in the nmse metric to encompass similarly located pixels as the one marked $\times$.

nfa median filter was at a higher structure size than that of the other nfa filters.

The results for the ball sequence, see figure 8.5(b), show a significant increase in nmse as the diameter increases from 1 to 3 pixels. A similar phenomenon is observable for the pendulum sequence, figure 8.5(a), but for the nfa discontinuity filter only. As the structure sizes increase further, the gap between the fa and nfa filters widens. For the ball sequence, the nmse seems to grow for very large structures.

8.1.3.2 Discussion

The gap between the fa and nfa filters widens with larger structure sizes because fairly large structures are needed, in high-noise environments, to properly estimate the flow fields used by the fa filters. All the filters, including the nfa, benefit from the increase in temporal stability caused by larger structure sizes. The additional gain when enlarging the structures is obvious for filters of spatial extent. The median filter is particularly sensitive of structure sizes, accounting for the prolonged reduction of nmse in the plots.

The increase in nmse for the ball sequence as the diameter increases from 1 to 3 pixels can be explained by the following: When the size of the ball is merely a single pixel, there are eight pixels included in the nmse that are background and one pixel that is foreground. Any filter reducing, or completely removing, the foreground pixel will still have a very low nmse. However, when the structure grows,

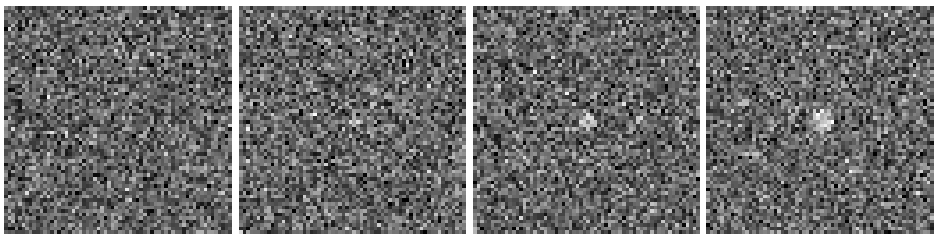


Figure 8.4: Small structures in noisy environments. The images are from the moving ball sequence with ball diameters, from left to right: 1,3,5 and 7 pixels. The structures with diameters 1 and 3 are hard to detect even in running video.

the ratio of foreground and background pixels even out, yielding the more correct, increased nmse. The increase for the nfa discontinuity filter, while not for the other nfa filters, in figure 8.5(a), is due to the fact that even though the discontinuity filter will remove, or severely dampen, the one-pixel pendulum at its high-motioned tip, the filter will give excellent results at the more motion stable pendulum top. The other filters, on the other hand, will not have the benefit of any excellent results anywhere, hence the high nmse for both the average and median filters. For the two-pixel pendulum, the discontinuity filter still removes, or heavily dampens, the pendulum tip, but the benefit of the foreground-background proportion bias in the nmse is no longer present.

The slight growth in nmse for very large structures can be accounted for by the increase in *normal* as opposed to *true* flow that is an unavoidable consequence of the energy-based approach and its failure to handle the *aperture problem*. The benefits of increased structure size suppress this phenomenon for the other filters.

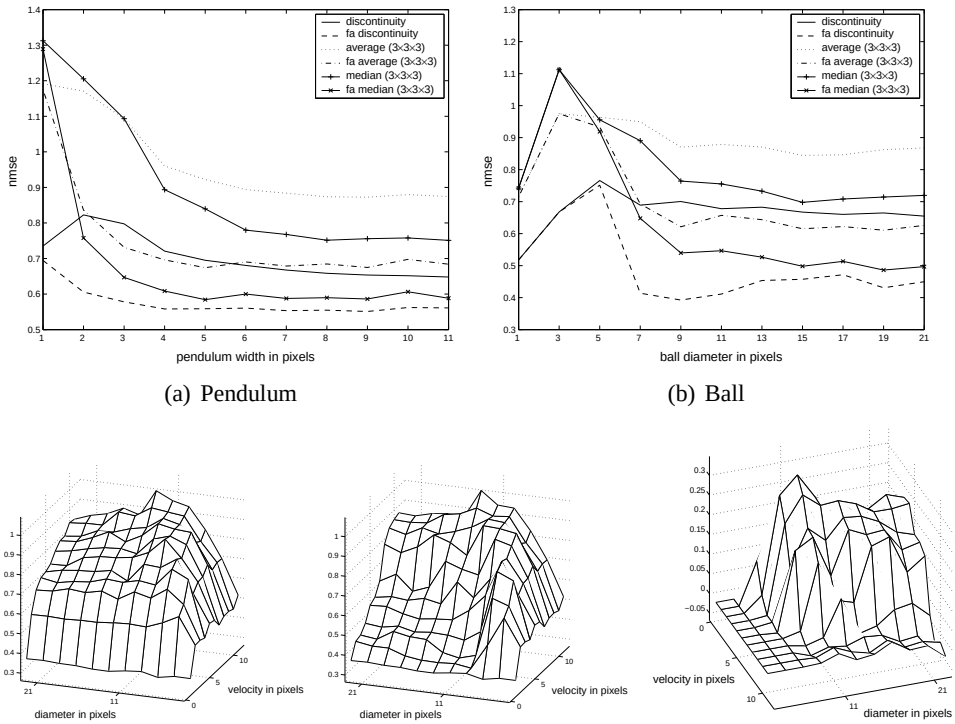
8.1.4 Smooth images

The flow-adaptive filters were tested on image sequences consisting of moving structures with both step- and smooth edges.

8.1.4.1 Results

Generally the filters with more than one pixel spatial extent had a much lower nmse for the smoothed sequences than the strictly temporal discontinuity filter. The overall best filter in the smoothed case was the average filter. Only the Gaussian filter could match the discontinuity filter for the step-edged sequences.

Table 8.1 shows that for the wall sequence, there is merely a negligible difference between the nmse for the two sequences using the discontinuity filter. Further, the average filter is the only filter which has a worse performance for the step-edge



(a) Pendulum (b) Ball

(c) Nfa (d) Fa (e) Nfa minus Fa

Figure 8.5: The effects of varying the size of the image structures. Except for the size parameters, *standard parameters* were used in the generation of the sequences. a)-b) The nmse for the pendulum and ball sequences for varying structure sizes. c) The nmse for the nfa discontinuity filter using the ball sequence with varying velocity and structure sizes. d) as in c) only using the fa discontinuity filter. d) The difference between c) and d). Note the change in view angle.

Filter	Edges	Wall	Annulus	Pendulum	Ellipse
Discont.	step	0.3683	0.5569	0.5515	0.5666
	smooth	0.3662	0.4088	0.4024	0.4116
Average	step	0.7426	0.6963	0.6769	0.6617
	smooth	0.1206	0.1469	0.1327	0.1311
Median	step	0.4870	0.5993	0.5808	0.5613
	smooth	0.1437	0.1672	0.1536	0.1527
Gaussian	step	0.3816	0.5033	0.4855	0.4730
	smooth	0.1362	0.1500	0.1437	0.1427

Table 8.1: The effects in nmse of smoothing the edges of the sequences before adding the noise. *Standard parameters* for the sequences were used. The filters were implemented using the flow-adaptive approach, and the average, median and Gaussian filters had an extent of $3 \times 3 \times 3$ pixels. Note that the effective metric regions, pixels deemed as edges, are increased in the smoothed case.

wall sequence compared to the other sequences. For the smooth images, all filters have a lower nmse for the wall sequence than the others.

8.1.4.2 Discussion

The filters of spatial extent benefit from the increased spatial stability of the pixel intensities, that is, the filters have low-pass characteristics which better fit the smooth image model. For the additive Gaussian noise model, the average filter is the locally maximum likelihood, accounting for the impressive results for that filter in the smoothed case. The more centrally weighted Gaussian-shaped filter retains sharp transitions better than the average filter, and performs better than the edge-preserving median filter due to the choice of a Gaussian additive noise model, which is better for the Gaussian filter, and the high-penalized erosion defect of the median.

All the sequences, except for that depicting a moving wall, gain structure sizes by the smoothing operation. The increase in structure size improves the flow-estimation and reduces the temporal instability, yielding the improved nmse for the discontinuity filter for all but the wall sequence. The high nmse for the average filter on the wall sequence can be explained by the principles illustrated in figure 8.3(b): Additional pixels, which are better handled by the filter, are included in the nmse for the other sequences. Such an effect is not seen for the median and Gaussian filters because of the other obvious advantages they have in the wall sequence.

8.1.5 Metric region and sizes of the non-adaptive filters

The variance-detection filter used in the error metric, the one setting S in equation (7.4), was enlarged to a spatial extent of 9×9 pixels. Such a filter size dilates the effective metric regions, making filtering artifacts like motion blurring affect the nmse considerably. Motion blur resulting from discontinuity filtering is discussed in section 8.1.6. The appendix contains additional data related to this section.

8.1.5.1 Results

The fa filters had a nmse substantially lower than their nfa counterparts, and the gap widened as the temporal extent increased. All the fa filters with a 3 pixel temporal extent had lower nmse than their strictly spatial counterparts, and remarkably lower nmse than the 3-pixel spatial extent nfa filters. Increasing the filter sizes to 5×5 pixels for the strictly spatial filters had a slight positive effect on the nmse for both the median and Gaussian filters, but the average filter had in that case an nmse increase. When the temporal sizes were extended, the larger 5×5 filters performed worse than their 3×3 counterparts.

In table 8.2, which shows the nmse results for the wall sequence, the fa filters display improvement as the temporal extents increase all the way to 7 pixels.

8.1.5.2 Discussion

The results show that the fa filters prevail over the nfa filters, with a significant reduction of motion blur and increased filter stability. When the temporal sizes increase beyond 3 pixels, the filters get more prone to flow-estimation inaccuracies, which accounts for the slight increase in nmse for such filters in some of the sequences with complex movement. The central weighting of the Gaussian filter and the edge-preserving properties of the median explain why those filters improve while the average worsens when increasing the spatial extent.

8.1.6 Leakage parameter

The leakage parameter of the discontinuity filter has a major impact on the filtering results. Figure 8.6 shows the nmse plots for the annulus sequence, together with examples of images. The other sequences gave similar results.

8.1.6.1 Results

The nmse plot of the nfa discontinuity filter in figure 8.6(a) has a convex shape starting with a sharp decline, and entering a minimum at a leakage of 0.3. The fa plot shows an overall lower nmse, and a prolonged decline not reaching minimum

Filter	Flow adaptive	Temporal extent 1	3	5	7
Average 3×3	no	0.3195	0.4677	0.6920	0.8675
	yes		0.2487	0.2350	0.2302
Median 3×3	no	0.3112	0.3932	0.5381	0.6699
	yes		0.2007	0.1772	0.1690
Gaussian 3×3	no	0.2862	0.3279	0.4723	0.6097
	yes		0.1868	0.1572	0.1451
Average 5×5	no	0.3839	0.4395	0.6603	0.8480
	yes		0.3610	0.3534	0.3514
Median 5×5	no	0.3074	0.3000	0.4705	0.6236
	yes		0.2627	0.2538	0.2507
Gaussian 5×5	no	0.2756	0.3519	0.4796	0.6100
	yes		0.2411	0.2306	0.2256

Table 8.2: The effects in nmse of varying the temporal extent of the filters when filtering the wall sequence with a velocity of 2 pixels per frame. Note that the variance-detection filter was set to 9×9 pixels to detect possible motion blur.

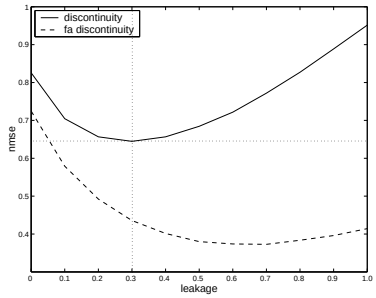
until a leakage of approximately 0.7.

Figure 8.6(c) and 8.6(d) show that the noise reduction for zero leakage is limited. With a leakage of 0.3 the noise reduction is more pronounced, with only slight signs of motion blurring for the nfa filter. When the leakage is set to 1.0 the noise is reduced substantially, but the image quality is severely degraded by motion blurring for the nfa filter. The fa filter is virtually free of such blurring, and only modest smoothing of the edges is introduced.

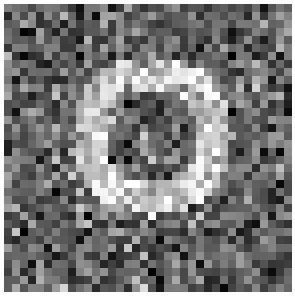
8.1.6.2 Discussion

The increased leakage parameter improves the damping of the noise, but when set too high, it causes motion blur for the nfa filter. The fa filter, on the other hand, has an *inborn* mechanism for handling image structural movement, and the leakage parameter can thus dampen the noise without the introduction of motion blur.

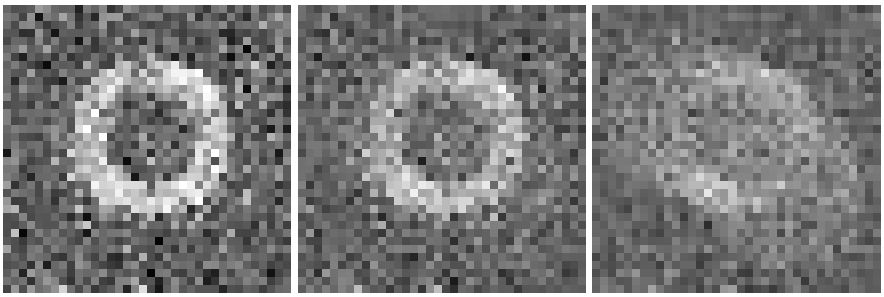
Note that the minimum nmse for the nfa filter in figure 8.6(a) justifies the choice of a leakage parameter of 0.3 introduced in section 7.2.



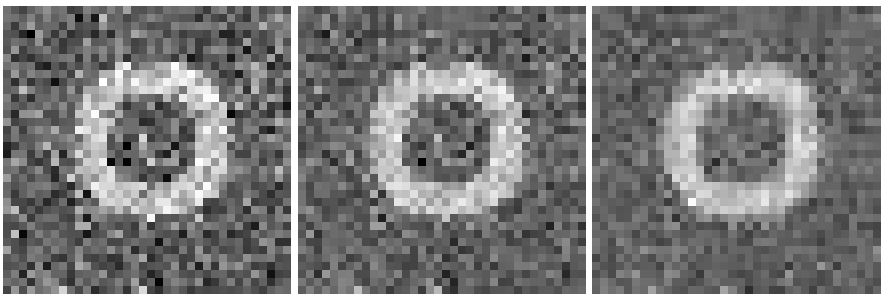
(a)



(b) Original



(c) Nfa, leakage: 0, 0.3 and 1.0



(d) Fa, leakage: 0, 0.3 and 1.0

Figure 8.6: a) The nmse when varying the leakage parameter from 0 to 1.0 for the annulus sequence with *standard parameters*. b)-c) Example images.

8.1.7 Tensor-based filter

In this section the sizes of the image structures were increased to better match the larger energy-based filters. The variance-detection filter was set to 9×9 pixels to give a fair error metric regarding both the size and the nature of the tensor-based filters. The α parameter in equation (7.1) was obtained by minimizing the nmse for the nfa tensor-based filter on the enlarged annulus sequence used in figure 8.7(a) with a per-frame angle increment of $\frac{\pi}{16}$.

8.1.7.1 Results

Generally the fa tensor-based filter showed an improvement in both nmse and visual appearance over the nfa twin filter. The fa filter outputs had less noise and sharper edge transitions than the nfa outputs.

Figure 8.7(a), which shows the nmse results for the enlarged annulus sequence, reveals a noticeable lower nmse for the fa tensor-based filter than the nfa counterpart. Further, the fa $3 \times 3 \times 3$ Gaussian-shaped filter has a lower nmse than the nfa tensor-based filter for a large velocity interval. When the movement of the annulus becomes substantial, the median filters have a more rapid nmse increase than the others. At extreme velocities, the tensor-based filters approach a steady-state lower than the other filters.

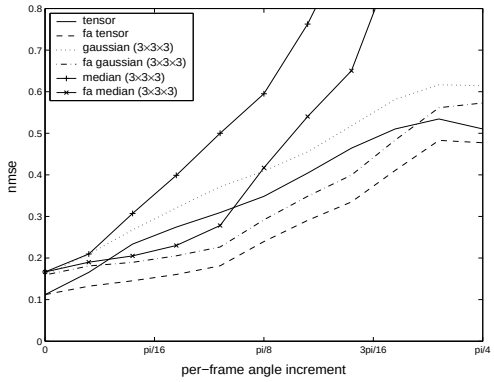
The images in figure 8.7 show that in regions where the gradient is in the direction of motion, the nfa tensor-based filter has a worse visual appearance than that of its fa twin. The images also reveal that the smoothing of homogeneous regions is better for the tensor-based filters than that of the Gaussian-shaped.

8.1.7.2 Discussion

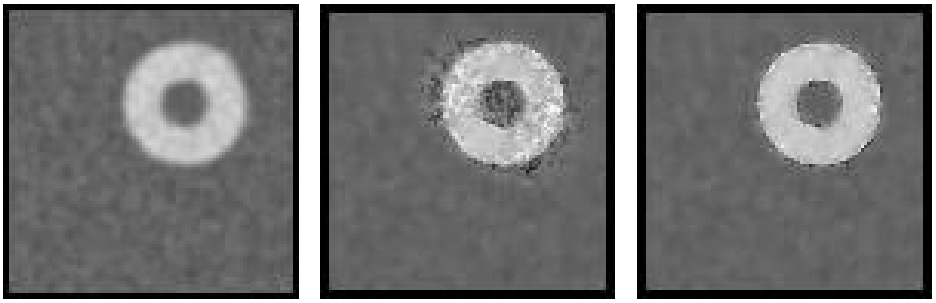
The noticeable improvement in both nmse and visual appearance for the fa filter confirms the prevalence of the fa approach even in the case of complex, adaptive filters.

The low nmse results for the fa Gaussian-shaped filter indicates that even simple isotropic filters can, using the flow-adaptive approach, compete with advanced adaptive filters. A general conclusion that the Gaussian-shaped filter outperforms the adaptive tensor-based filter can, of course, not be made based on this experiment. The poor results for the median filters for high velocities is due to the erosion property, although in the temporal direction. At extreme velocities, the tensor-based filters still manage to align themselves spatially, accounting for the low nmse for those filters.

The poor performance where the gradient is in the direction of motion for the nfa tensor-based filter is because of both the non-linear path of motion and the extent



(a) Enlarged annulus



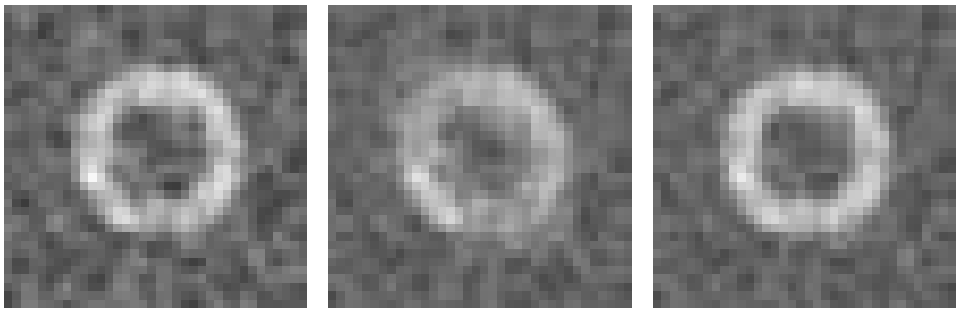
(b) Gaussian, fa (c) Tensor-based, nfa (d) Tensor-based, fa

Figure 8.7: a) The nmse results for varying the velocity in an enlarged annulus sequence. b)-d) Example images from the enlarged annulus sequence. Each frame is 96×96 pixels, and the inter-frame angle increment is $\frac{\pi}{16}$.

of the directed high-pass filters in their non-radial orientation, that is, when aligning the filter along the orientation of the motion, the filter will necessarily allow some of the high-pass content in the temporal direction to pass as well. In the fa approach the *transformation* to a motionless state removes such errors. For homogeneous regions, the size of the tensor-based filter yields the improved smoothness over the smaller Gaussian-shaped filter.

8.1.8 Visual inspection

Visual inspection of the filtered images reveals that the flow-adaptive filters give images with less motion blur compared to the non-adaptive filters. Figure 8.6 and 8.8 show examples of filtered images. More examples are found in the appendix. It is utmost likely that any quality metric based on visual perception would rank the fa above the nfa filters. Figure 8.8 contains in addition a strictly spatial filter, revealing that the filters with more than one pixel temporal extent are better at damping the noise.



(a) Spatial 3×3 (b) Nfa $3 \times 3 \times 3$ (c) Fa $3 \times 3 \times 3$

Figure 8.8: Examples of images filtered with the Gaussian-shaped convolution filter. The movement of the annulus for the example frames are downwards to the right.

8.2 Real ultrasound images

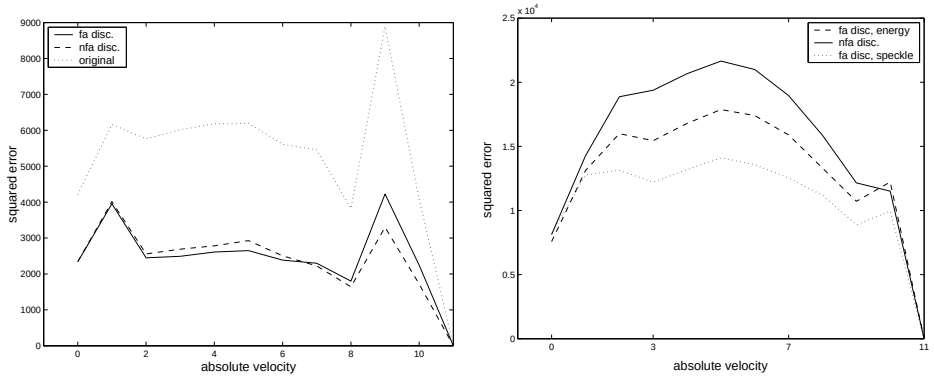
The real ultrasound images were filtered using the Gaussian-based convolution, the median and the discontinuity filter, although the focus was on the latter. The following web page contains all the filtered image sequences included in this thesis: *http://folk.uio.no/arej/ultrasoundSequences*

8.2.1 Results

The flow-adaptive filter outputs had reduced signs of motion blur compared to the non-flow-adaptive filter outputs. The moving structures in the noisy images were also more visible in the fa case in that the contrast quality was improved. Figure 8.10 shows examples of images from a noisy sequence.

Figure 8.9(a) shows how well local planes fit the discontinuity filtered data, a metric measuring smoothness as described in section 7.4.2, for a noisy sequence. Local planes fit the fa better than the nfa outputs for velocities below 7 pixels per frame. At higher velocities, the graph shows the opposite.

Figure 8.9(b) shows how well the local image structures are retained using the metric described in section 7.4.2. The results when the flow is estimated by speckle tracking are also included. The fa versions have less error for all the velocities, except for the highest absolute velocity where the energy-based filter is approximately equal to the nfa filter. At zero velocity, the fa filters are slightly better than their nfa counterpart. The fa filter using flow estimated by speckle tracking has less error than the fa filter using flow estimated using the energy-based approach for all velocities.



(a) Plane fitting, noisy sequence (b) Local difference, clean sequence

Figure 8.9: Results when using the metric described in section 7.4.2 on real ultrasound images filtered with the discontinuity filter. Note that results when the flow is estimated based on speckle tracking is included in b).

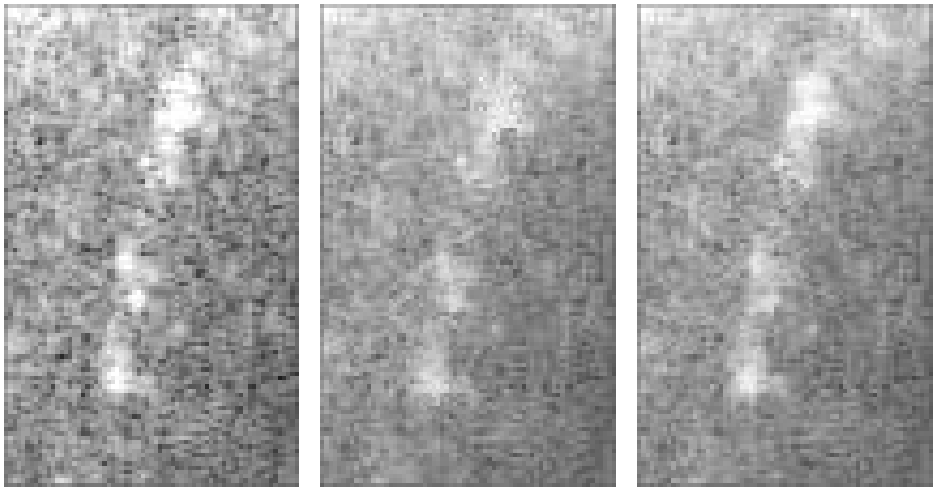
8.2.2 Discussion

The obvious improvement when visually inspecting the images and the graphs in figure 8.9 show that the flow-adaptive approach triumphs in the domain of real ultrasound images as well.

The sharp rise for the fa filter in figure 8.9(a) can be explained by the fact that the fa filter better retains such high-velocity neighborhoods rather than smearing them out. It should be noted that the number of neighborhoods with an estimated velocity of more than 7 pixels per frame is less than on fiftieth the number of neighborhoods with a velocity between 1 and 7.

The results in figure 8.9(b) show that the fa filters retain the original structures in nearly noiseless images better than the nfa filter. The lower error for the filter based on speckle tracking indicates that such a method is superior at flow estimation in nearly noiseless sequences. However, as the noise level increased, the speckle tracking results quickly deteriorated, and in cases of considerable noise, the energy-based methods had to be used. In a practical setting, a measure of local variance of the resulting flow fields could be used to guide the choice of using either a template-matching or an energy-based approach.

To explain the difference for zero velocity in figure 8.9(b), it must be noted that the error measure is taken over a window and hence the metric incorporates pixels besides that of the center. The other pixels could very well belong to neighborhoods of motion.



(a) Original (b) Nfa (c) Fa

Figure 8.10: Magnified extracts of ultrasound images filtered with the discontinuity filter

Chapter 9

Conclusion

This thesis has addressed the problem of unwanted motion blurring of images in three-dimensional filtering, and introduced a novel filtering approach termed the general flow-adaptive filter. The principle of the approach is to spatially adapt the entire filter lattice to possibly complex spatial movements in the temporal domain by incorporating local flow-field estimates.

Two adaptive and three non-adaptive filters were implemented using the flow-adaptive approach. Their performance was tested on both synthetic, and real ultrasound, image sequences. The synthetic sequences consisted of moving structures with varying size, velocity and edge smoothness, and were corrupted by several different noise models. The ultrasound images were of beating hearts.

An edge-adaptive normalized mean-square error was used as the metric for the synthetic sequences, and the error was severely reduced using the flow-adaptive technique, as much as halved in many instances. There were even indications that a simple Gaussian-shaped convolution filter could outperform larger and more complex adaptive filters by implementing the flow-adaptive procedure. For the ultrasound image sequences, the filters which adopted the flow-adaptive principles had outputs with less motion blur and sharper contrast compared to the outputs of the non-flow-adaptive filters.

At the cost of flow estimation, the flow-adaptive approach substantially improved the performance of all the filters.

9.1 Suggestions for further study

The most obvious extension to what is presented in this thesis would to test a wider range of filters for improvement using the flow-adaptive approach. However, the filters included in this thesis cover quite wide categories, and it is unlikely that other filters would deviate in that they would not benefit from the flow-adaptive

approach. On the other hand, real-life image domains besides that of the ultrasound could be an interesting study.

The flow-field estimation could become better. In the energy-based case, the use of spatial course-to-fine pyramids would improve the stability and also raise the upper velocity constraints found in the single-scale approach [24]. The speckle tracking would benefit from a per-sequence choosing of window sizes. Although computational efficiency has not been an issue in this thesis, both approaches have ways to reduce computational complexity, as mentioned in chapter 6. Other methods of estimating the flow fields could also be explored.

One idea to further improve the general flow-adaptive filter would be to let its temporal extent be adaptively set based on both intensity statistics, like in the discontinuity filter, and the velocity estimation certainties.

Bibliography

[1] L. Alparone, M. Barni, F. Bartolini, and V. Cappellini. Adaptively weighted vector-median filters for motion-fields smoothing. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2267–2270, 1996.

[2] Bjørn A. J. Angelsen. *Ultrasound Imaging: Waves, Signals and Signal Processing*. Department of Biomedical Engineering, NTNU, Norway, 1996.

[3] Andreas Austeng and Sverre Holm. Sparse 2-d arrays for 3-d phased array imaging – design methods. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 49:1073–1086, 2002.

[4] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr. A generalization of median filtering using linear combinations of order statistics. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 31:1342–1350, 1983.

[5] T. George Campbell and J.M.H du Buf. A quantitative comparison of median-based filters. In *Visual Communications and Image Processing*, volume 1360, pages 176–187. SPIE, 1990.

[6] C. Chinrungrueng and A. Suvichakorn. Fast edge-preserving noise reduction for ultrasound images. *IEEE Transactions on Nuclear Science*, 48:849–854, 2001.

[7] C.H. Chu. The application of an adaptive plan to the configuration of nonlinear image processing algorithms. In Edward J. Delp, editor, *Nonlinear Image Processing*, volume 1247, pages 248–257. SPIE, 1990.

[8] T.R. Crimmins. Geometric filter for speckle reduction. *Applied Optics*, 24:1438–, 1985.

[9] R.N. Czerwinski, D.L. Jones, and W.D. O'Brien. Ultrasound speckle reduction by directional median filtering. In *International Conference on Image Processing*, volume 1, pages 358–361, 1995.

[10] Per-Erik Danielsson. Getting the median faster. *Computer Graphics and Image Processing*, 17:71–78, 1981.

[11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[12] A.N. Evans and M.S. Nixon. Biased motion-adaptive temporal filtering for speckle reduction in echocardiography. *IEEE Transactions on Medical Imaging*, 15:39–50, 1996.

[13] B.H. Friemel, L.N. Bohs, and G.E. Trahey. Relative performance of two-dimensional speckle-tracking techniques: normalized correlation, non-normalized correlation and sum-absolute-difference. In *IEEE Ultrasonics Symposium*, volume 2, pages 1481–1484, 1995.

[14] Gösta H. Granlund and Hans Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995.

[15] T. Greiner, C. Loizou, M. Pandit, J. Mauruschat, and F.W. Albert. Speckle reduction in ultrasonic imaging for medical applications. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 2993–2996, 1991.

[16] Robert M. Haralick and Linda G. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1992.

[17] N.R Harvey, S. Marshall, and G. Matsopoulos. Adaptive stack-filters towards a design methodology for morphological filters. In *IEEE Colloquium on Morphological and Nonlinear Image Processing Techniques*, pages 6/1–6/4, 1993.

[18] A. Hoess and H. Ermert. Adaptive wiener filtering for b-mode image improvement. In *IEEE Ultrasonics Symposium*, volume 2, pages 1219–1222, 1992.

[19] J. Hu and X. Hu. Application of median filter to speckle suppression in intravascular ultrasound images. In *Second Australian and New Zealand Conference on Intelligent Information Systems*, pages 302–306, 1994.

[20] J.J. Huang and E.J. Coyle. Perceptual error criteria and restoration of images with stack filters. In *IEEE International Conference on Image Processing*, 1998.

[21] T. S. Huang, G. J. Yang, and G. Y. Tang. A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 27, No. 1:13–18, 1979.

[22] G.B. Adams III, E.J. Coyle, L. Lin, L.E. Lucke, and K.K. Parhi. Input compression and efficient VLSI architectures for rank order and stack filters. *Signal Processing*, 38(3):441–453, 1994.

[23] B.I. Justusson. *Median Filtering: Statistical Properties in Two-Dimensional Digital Signal Processing*. Springer Verlag, 1981.

[24] Jörgen Karlholm. *Efficient Spatiotemporal Filtering and Modelling*. PhD thesis, Linköping University, Sweden, 1996.

[25] J.W. Klingler, C.L. Vaughan, T.D. Fraker, and L.T. Andrews. Segmentation of echocardiographic images using mathematical morphology. *IEEE Transactions on Biomedical Engineering*, 35:925–934, 1988.

[26] Hans Knutsson. *Filtering and Reconstruction in Image Processing*. PhD thesis, Linköping University, Sweden, 1982.

[27] Hans Knutsson. Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, June 1989.

[28] Hans Knutsson, Mats Andersson, and Johan Wiklund. Advanced filter design. In *The 11th Scandinavian Conference on Image Analysis*, June 1999.

[29] C. Kotropoulos, X. Magnisalis, I. Pitas, and M.G. Strintzis. Nonlinear ultrasonic image processing based on signal-adaptive filters and self-organizing neural networks. *IEEE Transactions on Image Processing*, 3:65–77, 1994.

[30] H. Lester and S.R. Arridge. A survey on hierarchical non-linear medical image registration. *Pattern Recognition*, 32:129–149, 1999.

[31] Quan Liang. *Boundary Detection in Cardiovascular Ultrasonic Images Based on Multiscale Dynamic Programming*. PhD thesis, Chalmers University of Technology, Sweden, 1999.

[32] J.H. Lin and E.J. Coyle. Generalized stack filters and minimum mean absolute error estimation. In *IEEE International Symposium on Circuits and Systems*, pages 2799–2802, 1988.

[33] J.H. Lin, T.M Sellke, and E.J. Coyle. Adaptive stack filtering under the mean absolute error criterion. In Edward J. Delp, editor, *Nonlinear Image Processing*, volume 1247, pages 182–193. SPIE, 1990.

[34] C. Loizou, C. Christodoulou, C.S. Pattichis, R. Istepanian, M. Pantziaris, and A. Nicolaides. Speckle reduction in ultrasound images of atherosclerotic carotid plaque. In *14th International Conference on Digital Signal Processing*, volume 2, pages 525– 528, 2002.

[35] William Menke. *Geophysical Data Analysis: Discrete Inverse Theory*. Academic Press, 1989.

[36] Sanjit K. Mitra. *Digital Signal Processing: A Computer-Based Approach*. McGraw-Hill, second edition, 2001.

[37] David S. Moore and George P. McCabe. *Introduction to the Practice of Statistics*. W.H. Freeman & Company, third edition, 1998.

[38] Bjørn Olstad. Noise reduction in ultrasound images using multiple linear regression in a temporal context. In *Electronic Imaging. Science & Technology, San Jose*. SPIE/SPSE, 1991.

[39] Francesco Palmieri and R. Edward Croteau. Hybrid order statistic filters for adaptive image restoration. In Edward J. Delp, editor, *Nonlinear Image Processing*, volume 1247, pages 40–50. SPIE, 1990.

[40] Francesco Palmieri and R. Edward Croteau. Image restoration based on perception-related cost functions. In Edward R. Dougherty, Gonzales R. Arce, and Charles G. Boncelet, editors, *Nonlinear Image Processing II*, volume 1451, pages 24–35. SPIE, 1991.

[41] Ioannis Pitas. *Digital Image Processing Algorithms and Applications*. John Wiley & Sons, 2000.

[42] K.N. Plataniotis, D. Androutsos, and A.N. Venetsanopoulos. Vector directional filters: An overview. In *IEEE Canadian Conference on Electrical and Computer Engineering*, pages 106–109, 1997.

[43] William K. Pratt. *Digital Image Processing*. John Wiley & Sons, third edition, 2001.

[44] Jens U. Quistgaard. Ultrasonic image formation: implications for the image processing practitioner. In *IEEE International Conference on Image Processing*, volume 3, pages 533–537, 1994.

[45] Jens U. Quistgaard. Signal acquisition and processing in medical diagnostic ultrasound. *IEEE Signal Processing Magazine*, 14:67–74, 1997.

[46] John A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, second edition, 1994.

[47] M.A. Sapia, M.D. Fox, L.M. Loew, and J.C. Schaff. Ultrasound image deconvolution using adaptive inverse filtering. In *12th IEEE Symposium on Computer-Based Medical Systems*, pages 248–253, 1999.

[48] A. Schistad and T. Taxt. Speckle reduction in ultrasound images using temporal and spatial context. In *Nuclear Science Symposium and Medical Imaging Conference*, volume 3, pages 2210–2214, 1991.

[49] Robert L. Solso. *Cognitive Psychology*. Allyn & Bacon, sixth edition, 2000.

[50] R. N. Strickland and D. Chang. An adaptable edge quality metric. In *Visual Communications and Image Processing*, volume 1360, pages 982–995. SPIE, 1990.

[51] Changming Sun. Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *International Journal of Computer Vision*, 47(1/2/3):99–117.

[52] Harry L. Van Trees. *Optimum Array Processing (Detection, Estimation, and Modulation Theory, Part IV)*. John Wiley & Sons, 2002.

[53] Mark Allen Weiss. *Data Structures and Algorithm Analysis*. Addison-Wesley, second edition, 1994.

[54] P. D. Wendt, E. J. Coyle, and N. C. Gallagher. Stack filters. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34, No. 4:898–911, 1986.

[55] Carl-Fredrik Westin. *A Tensor Framework for Multidimensional Signal Processing*. PhD thesis, Linköping University, Sweden, 1994.

[56] J. Yoo, K.L. Fong, J.J. Huang, E.J. Coyle, and G.B. Adams III. A fast algorithm for designing stack filters. *IEEE Transactions on Image Processing*, 8:1014–1028, 1999.

[57] J. Yoo, K.L. Fong, G.B. Adams III, and E.J. Coyle. Stack filters: theory and applications. In *IEEE Twenty-Seventh Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1006–1010, 1993.

Appendix A

Additional Plots, Tables and Images

This appendix contains additional performance plots and tables together with more examples of filtered images.

Figure A.1 supplements section 8.1.1, while figure A.2 complements the figures in section 8.1.3. The tables A.1, A.2 and A.3 are related to section 8.1.5. Figure A.3 shows additional examples of filtered images.

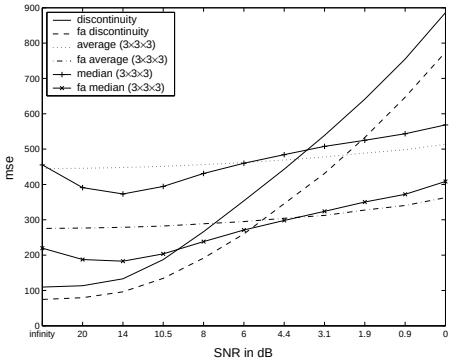
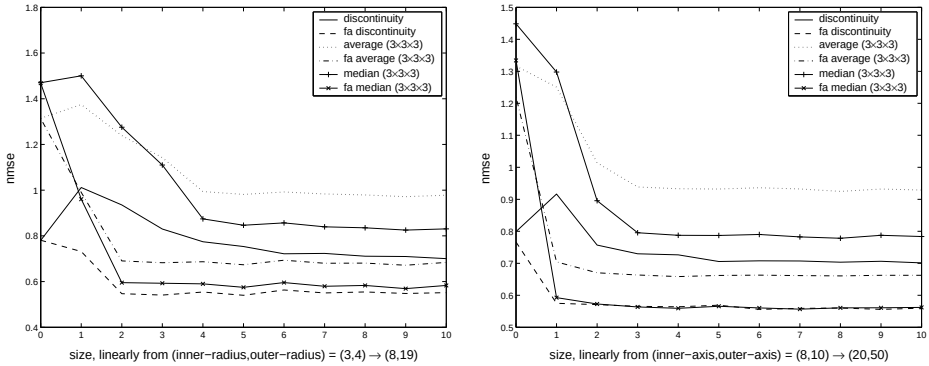


Figure A.1: Non-normalized mean squared-error metric using the 3×3 variance-detection filter with varying multiplicative noise on the circulating annulus sequence with *standard parameters*



(a) Annulus (b) Ellipse

Figure A.2: The nmse for the annulus and ellipse sequences for varying structure sizes. Except for the size parameters, *standard parameters* were used in the generation of the sequences.

Filter	Flow adaptive	Temporal extent 1	3	5	7
Average 3×3	no	0.3432	0.5817	0.8796	1.0385
	yes	0.3432	0.3059	0.3209	0.3475
Median 3×3	no	0.3427	0.5326	0.8901	1.0719
	yes	0.3427	0.2794	0.2930	0.3218
Gaussian 3×3	no	0.3164	0.3883	0.5872	0.7514
	yes	0.3164	0.2574	0.2652	0.2820
Average 5×5	no	0.4187	0.6342	0.9005	1.0385
	yes	0.4187	0.4175	0.4331	0.4542
Median 5×5	no	0.3500	0.5548	0.8992	1.0719
	yes	0.3500	0.3347	0.3550	0.3845
Gaussian 5×5	no	0.3184	0.4401	0.6274	0.7734
	yes	0.3184	0.3058	0.3191	0.3359

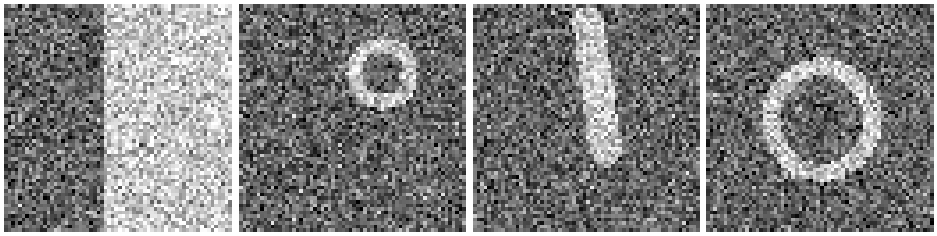
Table A.1: The effects in nmse of varying the temporal extent of the filters when filtering the annulus sequence (*standard parameters*.)

Filter	Flow adaptive	Temporal extent 1	3	5	7
Average 3×3	no	0.2994	0.3790	0.5522	0.7171
	yes	0.2994	0.2511	0.2632	0.2920
Median 3×3	no	0.3153	0.3284	0.5001	0.6979
	yes	0.3153	0.2378	0.2497	0.2819
Gaussian 3×3	no	0.2958	0.2881	0.3825	0.4901
	yes	0.2958	0.2232	0.2212	0.2361
Average 5×5	no	0.3312	0.4152	0.5848	0.7465
	yes	0.3312	0.3181	0.3345	0.3543
Median 5×5	no	0.2832	0.3337	0.5118	0.7079
	yes	0.2832	0.2603	0.2788	0.3045
Gaussian 5×5	no	0.2696	0.3080	0.4057	0.5125
	yes	0.2696	0.2469	0.2560	0.2696

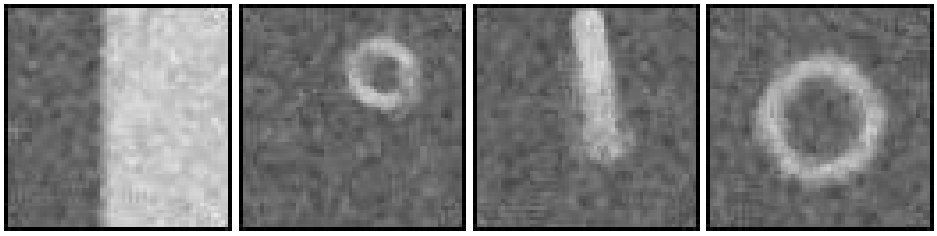
Table A.2: The effects in nmse of varying the temporal extent of the filters when filtering the pendulum sequence (*standard parameters*.)

Filter	Flow adaptive	Temporal extent 1	3	5	7
Average 3×3	no	0.3269	0.4383	0.7160	0.9395
	yes	0.3269	0.2806	0.3095	0.3546
Median 3×3	no	0.3278	0.3663	0.6654	0.9329
	yes	0.3278	0.2582	0.2963	0.3554
Gaussian 3×3	no	0.3097	0.3263	0.4696	0.6238
	yes	0.3097	0.2398	0.2499	0.2764
Average 5×5	no	0.3885	0.4949	0.7381	0.9471
	yes	0.3885	0.3764	0.3961	0.4281
Median 5×5	no	0.3225	0.3798	0.6607	0.9313
	yes	0.3225	0.2982	0.3262	0.3734
Gaussian 5×5	no	0.3038	0.3635	0.4981	0.6474
	yes	0.3038	0.2830	0.2945	0.3177

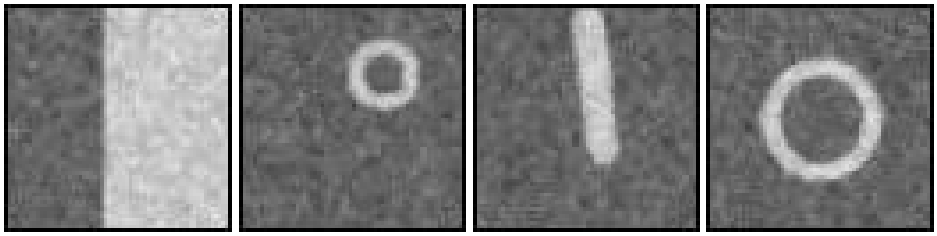
Table A.3: The effects in nmse of varying the temporal extent of the filters when filtering the ellipse sequence (*standard parameters*.)



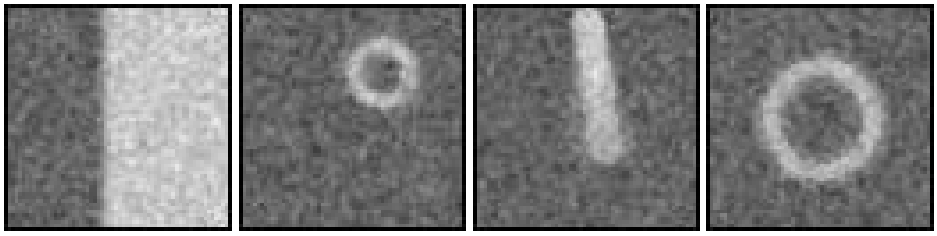
(a)



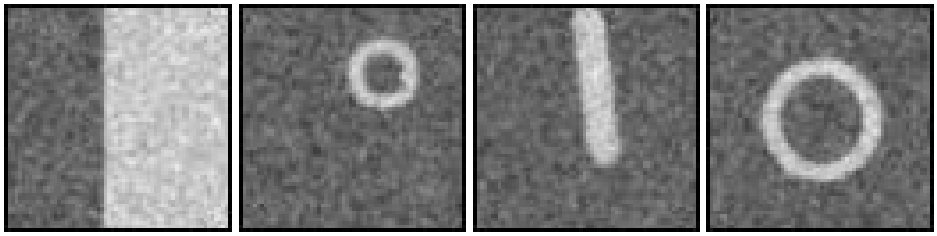
(b)



(c)



(d)



(e)

Figure A.3: Examples of filtered images. a) Original corrupted images. b)-c) Nfa and fa median filtered. d)-e) Nfa and fa Gaussian-shaped convolution filtered.

Appendix B

Mathematical Notations

$a, \vec{a}, \mathbf{A}$	notations for a scalar, a vector and a matrix (capital boldface)
$\vec{a}_k, \mathbf{A}_{kl}$	the kth element of the vector $\vec{a}$, and the element in the intersection between the kth row and the lth column of the matrix $\mathbf{A}$.
$\vec{a}^t, \mathbf{A}^t$	the transpose of the vector, $\vec{a}$, or matrix, $\mathbf{A}$
$\mathbf{A}^{-1}$	the inverse of the matrix $\mathbf{A}$
$\|\vec{a}\|$	the norm of the vector $\vec{a}$
$E[x^m]$	the mth moment of the random variable x
$\vec{a}_{()}$	the elements of a vector, $\vec{a}$, in ordered form, i.e., $\vec{a}_{()} = (a_{(1)}, \ldots, a_{(N)}), \quad a_{(1)} \leq a_{(2)} \leq \ldots \leq a_{(N)}$
$\vec{a} \cdot \vec{b}$	the scalar-product of the vectors $\vec{a}$ and $\vec{b}$
$\mathbf{T} \cdot \mathbf{U}$	the scalar-product of the tensors $\mathbf{T}$ and $\mathbf{U}$, i.e., $\mathbf{T} \cdot \mathbf{U} = \sum_{k,l} \mathbf{T}_{kl}\mathbf{U}_{kl}$
$\lambda_n, \vec{e}_n$	the nth largest eigenvalue and its corresponding eigenvector
f, F	a filter in spatial and Fourier space, respectively

Appendix C

Program-Code Excerpts

This appendix presents excerpts from the program code written for this thesis. The complete program consists of more than 2500 lines of Java code, and several hundred lines of code in the Matlab language.

C.1 Matlab code

C.1.1 Obtaining the quadrature filters

```
% OBTAIN A 3D QUADRATURE FILTER BY WEIGHTED MINIMUM SQUARED ERROR
%

M = 7;          % SIZE OF FILTER
N = 15;         % SAMPLING RESOLUTION

a = 2;
b = (1+5^.5);
c = (10+2*5^.5)^(-1/2);
refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[a,0,b]');
%refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[-a,0,b]');
%refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[b,a,0]');
%refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[b,-a,0]');
%refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[0,b,a]');
%refFunc = get3DQuadratureReferenceFunct(N, 2, 2^(-1.5), c*[0,b,-a]');

fprintf('generating weight matrix..');
% AS SUGGESTED IN Signal Processing for Computer Vision BY Granlund and Knutsson, 1995
wWin = zeros(N,N,N);
psi = 1.00;

for k1=0:N-1
  for k2=0:N-1
    for k3=0:N-1
      u = ( [k1; k2; k3]-floor(N/2) )/floor(N/2);
      p = norm(u);

      if p==0
        % GIVE DC-COMPONENT TOP PRIORITY:
        wWin(k1+1,k2+1,k3+1) = max(max(max(wWin)))*999;
      else
```

```
        wWin(k1+1,k2+1,k3+1) = p^(-1) + psi;
      end
    end
  end
end

fprintf('done\n');

fprintf('finding optimized filter..');
filt = getOptimized3DFilter(refFunc, M, wWin);
fprintf('done\n');

%
% PERFORMS THE WEIGHTED SQUARED ERROR MINIMIZATION
%

function [filt] = getOptimized3DFilter(refFunc, filtSize, weightWin)

N = length(refFunc);
refFunc = ifftshift(refFunc);
refFuncList = getVectorFrom3DMatrix(refFunc);

fprintf('generating DFT matrix..');
G = getDFT3DMatrixMiddle(N, filtSize);
fprintf('done\n');

w = getVectorFrom3DMatrix(ifftshift(weightWin));
W = diag(w);

fprintf('finding weighted least squares...');
f = inv(G'*W*G)*G'*W*refFuncList;
fprintf('done\n');

filt = get3DMatrix(f,filtSize,filtSize,filtSize);

filt = fftshift(filt);
```

C.2 Java Code

C.2.1 Creating the tensors

```
/*
* Makes a tensor field from the quadrature filter outputs.
* <quadAmps> contains the output-amplitudes of the six quadrature filters.
*/
public static float[][][][] getTensorField(float[][][] quadAmps) {
  int sizeY = quadAmps[0].length;
  int sizeX = quadAmps[0][0].length;

  float[][][][] tensorField = new float[sizeY][sizeX][3][3];

  float[][][] mk = getMMatrices();

  for (int x=0; x<sizeX; x++) {
    for (int y=0; y<sizeY; y++) {

      for (int k=0; k<6; k++) {
        for (int i=0; i<3; i++) {
          for (int j=0; j<3; j++) {
            tensorField[y][x][i][j] += quadAmps[k][y][x] * mk[k][i][j];
          }
```

```
        }
      }

    }
  }
  return tensorField;
}

/*
* Assembles the Mk matrices.
*/
private static float[][][] getMMatrices() {
  float[][][] mk = new float[6][3][3];

  float a = 2, b = (float)((1+Math.sqrt(5)));
  float c = (float)(1.0/Math.sqrt(10+2*Math.sqrt(5)));
  a *= c; b *= c;
  float[][] nk = new float[6][3];
  nk[0][0] = a;  nk[0][1] = 0;  nk[0][2] = b;
  nk[1][0] = -a; nk[1][1] = 0;  nk[1][2] = b;
  nk[2][0] = b;  nk[2][1] = a;  nk[2][2] = 0;
  nk[3][0] = b;  nk[3][1] = -a; nk[3][2] = 0;
  nk[4][0] = 0;  nk[4][1] = b;  nk[4][2] = a;
  nk[5][0] = 0;  nk[5][1] = b;  nk[5][2] = -a;

  for (int k=0; k<6; k++) {
    for (int i=0; i<3; i++) {
      for (int j=0; j<3; j++) {
        mk[k][i][j] = 5.0f/4*nk[k][i]*nk[k][j];
        if (i==j) mk[k][i][j] -= 1.0/4;
      }
    }
  }
  return mk;  // Mk = 5/4*nk*nk' - 1/4*I;
}
```

C.2.2 Extracting the flow

```
/*
* Extracts the flow from the tensors.
*/
public static float[][][] getFlowField(float[][][][] tensorField,
    float[][] flowFieldWeights, int QUADRATUREFILTERSIZE, float TENSORREGULARIZATION) {

  int sizeY = tensorField.length;
  int sizeX = tensorField[0].length;

  float[][][] flowField = new float[sizeY][sizeX][2];

  for (int x=QUADRATUREFILTERSIZE/2; x<sizeX-QUADRATUREFILTERSIZE/2; x++) {
    for (int y=QUADRATUREFILTERSIZE/2; y<sizeY-QUADRATUREFILTERSIZE/2; y++) {

      float[][] T = tensorField[y][x];
      // increase 'roundness', as described in [westin-phd] p116
      T[0][0] += TENSORREGULARIZATION;
      T[1][1] += TENSORREGULARIZATION;
      T[2][2] += TENSORREGULARIZATION;

      float[] eigs = Eigenvectors.getEigenvalues(T);

      float norm = Math.abs(eigs[0])+Math.abs(eigs[1])+Math.abs(eigs[2]);
```

```
      float[][] eigv = Eigenvectors.getEigenvectors(T,eigs);

      float p1 = eigs[2]>0 ? (eigs[2]-eigs[1])/eigs[2] : 0; // [granKnuts-sp] p254
      float p2 = eigs[2]>0 ? (eigs[1]-eigs[0])/eigs[2] : 0;
      float p3 = eigs[2]>0 ? eigs[0]/eigs[2] : 0;

      flowFieldWeights[y][x] = norm*(p2+0.01f);
      if (p3>p1 && p3>p2) flowFieldWeights[y][x] = 0;

      if (p1>p3 && p1>p2) // plane case
      {
        double denominator = eigv[0][2]*eigv[0][2] + eigv[1][2]*eigv[1][2];
        double velocityX = -eigv[2][2]*(eigv[0][2])/denominator;
        double velocityY = -eigv[2][2]*(eigv[1][2])/denominator;
        if (Math.abs(velocityX)<QUADRATUREFILTERSIZE
            & Math.abs(velocityY)<QUADRATUREFILTERSIZE) {
          flowField[y][x][0] = (float) ( velocityX );
          flowField[y][x][1] = (float) ( velocityY );
        }
      }

      if (p2>p3 && p2>p1) // line case
      {
        double velocityX = eigv[0][0]/eigv[2][0];
        double velocityY = eigv[1][0]/eigv[2][0];

        if (Math.abs(velocityX)<QUADRATUREFILTERSIZE
            & Math.abs(velocityY)<QUADRATUREFILTERSIZE) {
          flowField[y][x][0] = (float) ( velocityX );
          flowField[y][x][1] = (float) ( velocityY );
        }
      }

    }
  }
  return flowField;
}
```

C.2.3 Flow-adaptive convolution

```
/*
* Does a flow-adaptive convolution.
* If <flowBackwards>==null, single-way flow-adaption is used.
*/
public static float[][][] flowAdaptiveConvolve(float[][][] data, float[][][] filter,
                                    float[][][][] flow, float[][][][] flowBackwards) {

  int sizeZ = data.length;
  int sizeY = data[0].length;
  int sizeX = data[0][0].length;

  int filtSizeZ = filter.length;
  int filtSizeY = filter[0].length;
  int filtSizeX = filter[0][0].length;

  float[][][] out = new float[sizeZ][sizeY][sizeX];
  // will be filled with flow-adaptive data:
  float[][][] lattice = new float[filtSizeZ][filtSizeY][filtSizeX];

  for (int i=filtSizeZ/2; i<sizeZ-filtSizeZ/2; i++) {
    for (int j=filtSizeY/2; j<sizeY-filtSizeY/2; j++) {
      for (int k=filtSizeX/2; k<sizeX-filtSizeX/2; k++) {
```

```
        if (flowBackwards==null) fillFlowAdaptiveLattice(i,j,k,data,lattice, flow);
        else fillFlowAdaptiveLattice(i,j,k,data,lattice, flow, flowBackwards);

        for (int fi=0; fi<filtSizeZ; fi++) {
          for (int fj=0; fj<filtSizeY; fj++) {
            for (int fk=0; fk<filtSizeX; fk++) {
              out[i][j][k] +=
                filter[fi][fj][fk]*lattice[filtSizeZ-1-fi][filtSizeY-1-fj][filtSizeX-1-fk];
            }
          }
        }

      }
    }
  }

  return out;
}


/*
* Returns a 3D-matrix with the values of a flow-adaptive lattice.
* If flow leads outside image, the filling part is simply aborted.
*/
private static void fillFlowAdaptiveLattice(int i, int j, int k, float[][][] data,
                                            float[][][] lattice, float[][][][] flow) {
  int filtSizeZ = lattice.length;
  int filtSizeY = lattice[0].length;
  int filtSizeX = lattice[0][0].length;

  double tj = j;  // retain starting positions
  double tk = k;
  int pj = j;     // rounded-to-integer positions
  int pk = k;

  // the first half of the lattice
  for (int fi=0; fi<=filtSizeZ/2; fi++) {
    for (int fj=-filtSizeY/2; fj<=filtSizeY/2; fj++) {
      for (int fk=-filtSizeX/2; fk<=filtSizeX/2; fk++) {
        if ((pj+fj >= data[0].length) || (pk+fk >= data[0][0].length)
            || (pj+fj < 0) || (pk+fk < 0)) return;
        lattice[fi+filtSizeZ/2][fj+filtSizeY/2][fk+filtSizeX/2] = data[i+fi][pj+fj][pk+fk];
      }
    }
    tj += flow[i+fi][pj][pk][0];
    tk += flow[i+fi][pj][pk][1];
    pj = (int)Math.round(tj);
    pk = (int)Math.round(tk);
  }

  // now, go backwards
  tj = j - flow[i][j][k][0];
  tk = k - flow[i][j][k][1];
  pj = (int)Math.round(tj);
  pk = (int)Math.round(tk);

  for (int fi=-1; fi>=-filtSizeZ/2; fi--){
    for (int fj=-filtSizeY/2; fj<=filtSizeY/2; fj++) {
      for (int fk=-filtSizeX/2; fk<=filtSizeX/2; fk++) {
        if ((pj+fj >= data[0].length) || (pk+fk >= data[0][0].length)
            || (pj+fj < 0) || (pk+fk < 0)) return;
```

```
            lattice[fi+filtSizeZ/2][fj+filtSizeY/2][fk+filtSizeX/2] = data[i+fi][pj+fj][pk+fk];
          }
        }
        tj = tj - flow[i+fi][pj][pk][0];
        tk = tk - flow[i+fi][pj][pk][1];
        pj = (int)Math.round(tj);
        pk = (int)Math.round(tk);
      }
    }
```