Sequential Filter Trees for Efficient 2D 3D and 4D Orientation Estimation

Mats Andersson Johan Wiklund Hans Knutsson

LiTH-ISY-R-2070
1998

Sequential Filter Trees for Efficient 2D 3D and 4D Orientation Estimation

Mats Andersson Johan Wiklund Hans Knutsson
Computer Vision Laboratory
Linköping University, Sweden

Abstract

A recursive method to condense general multidimensional FIR-filters into a sequence of simple kernels with mainly one dimensional extent has been worked out. Convolver networks adopted for **2, 3 and 4D signals** is presented and the performance is illustrated for spherically separable quadrature filters. The resulting filter responses are mapped to a non biased **tensor representation** where the local tensor constitutes a robust estimate of both the shape and the orientation (velocity) of the neighbourhood.

A qualitative evaluation of this **General Sequential Filter concept** results in no detectable loss in accuracy when compared to conventional FIR (Finite Impulse Response) filters but the computational complexity is reduced several orders in magnitude. For the examples presented in this paper the attained speed-up is **5, 25 and 300 times** for 2D, 3D and 4D data respectively The magnitude of the attained speed-up implies that complex spatio-temporal analysis can be performed using standard hardware, such as a powerful workstation, in close to real time.

Due to the soft implementation of the convolver and the tree structure of the sequential filtering approach the processing is simple to reconfigure for the outer as well as the inner (vector length) dimensionality of the signal. The implementation was made in AVS (Application Visualization System) using modules written in **C**.

Contents

1 Introduction **1**

2 General Sequential Filters **2**

3 Optimization of Filter Coefficients **2**

4 Tensor Representation and Error Estimation **4**

5 A Filtering Structure for 2D Signals **6**

6 A Filtering Structure for 3D Signals **9**

7 A Filtering Structure for 4D Signals **12**

8 Implementation Aspects **15**

9 Acknowledgment **15**

A Sequential filters for 2D **16**

B Sequential Filters for 3D **17**

C Sequential Filters for 4D **18**

D References **20**

1 Introduction

In multidimensional signal processing, the main part of the computational power is usually spent on the initial filtering. The filters do for natural reasons need to be of the same 'dimensionality' as the signal and for 3D and 4D data, such as time sequences and time sequences of volumes, the number of filter coefficients increases dramatically and often limits the practical (interactive) use of the algorithm. In this paper (see also [9]) a new recursive method to optimize general multidimensional filters has been worked out which uses only a small fraction of the number of filter coefficients that is required by conventional filtering. The attained speed up implies that multidimensional signal analysis can be performed using standard hardware, such as a powerful workstation, in close to real time. An observation that is very important for the proposed approach is that the performance of a convolver, implemented in standard hardware is directly proportional to the number of kernel coefficients, but independent of the spatial coordinates for these coefficients. A software convolver will also be independent of built in limitations in special purpose hardware and can by simple means be reconfigured for signals of different dimensionality and vector length, see [10] for more details.

Using N coefficients it is always possible to find a best approximation to a given filter using all coefficients at once for a single filter. In many cases, however, a far more efficient way of attaining essentially the same filter is to use the same number of coefficients but distribute them over a number, (M), of smaller filters. These filters are then applied in sequence (sequential convolution) to obtain the final filter response.

The proposed method involves the following three steps:

1. Determine the number, M, of sequential filters that is appropriate for the present application.

2. Determine the distribution of the N coefficients, i.e. determine the coordinates of the coefficients for each of the M sequential filters.

3. Optimize the values of the N coefficients (distributed over the M filters) so that the combined effect of the filter sequence matches the reference filter as closely as possible.

To optimize these steps simultaneously is a complex problem indeed and an overall optimal solution has not been found, (and probably never will be). In order to reduce the complexity of the problem the number of filter components and the distribution of coefficient coordinates are performed by intuition and by taking advantage of the symmetry properties of the reference filter. Given the number of filter M and the spatial coordinates of each filter the last step is still cumbersome. A recursive method is proposed that in each step optimize each filter component in relation to the remaining $M-1$ components. Convergence of the algorithm is fast and typically only 4-6 iterations are needed. This sequential filter optimization method is applied to polar separable quadrature filters for 2D, 3D and 4D signals and a convolver tree structure and performance is presented.

2 General Sequential Filters

The objective of introducing sequential filtering is to approximate an ideal filter, $\check{F}(\mathbf{u})$, by M filter components, $F_k(\mathbf{u})$. In the Fourier domain this is expressed as:

$$\check{F}(\mathbf{u}) \approx F(\mathbf{u}) = \prod_{k=1}^{M} F_k(\mathbf{u}) \tag{1}$$

where $\mathbf{u}$ is the frequency variable. This operation is meaningful if the filters $F_k(\mathbf{u})$ can be implemented using a considerable smaller number of kernel coefficients in comparison to a direct ($M = 1$) implementation of $F(\mathbf{u})$. In the spatial domain eq. (1) corresponds to:

$$f(\boldsymbol{\xi}) = f_1(\boldsymbol{\xi}) * f_2(\boldsymbol{\xi}) * \ldots * f_M(\boldsymbol{\xi}) \tag{2}$$

where $\boldsymbol{\xi}$ are spatial coordinates. The filter function $F_k(\mathbf{u})$ of a convolution kernel with N_k coefficients is given by its Fourier transform

$$F_k(\mathbf{u}) = \sum_{n=1}^{N_k} f_k(\boldsymbol{\xi}_n)\ \exp(-i\,\boldsymbol{\xi}_n \cdot \mathbf{u}) \tag{3}$$

where $\boldsymbol{\xi}_n$ define the coordinates where $f_k(\boldsymbol{\xi}) \neq 0$.

Note that so far no mention has been made on how these coordinates are chosen. This decision depend on the number of filter components M as well as the total number of kernel coefficients N. In most cases the symmetry of $F(\mathbf{u})$ reduce the number of possibilities to a manageable level. The coefficients of a filter component can for example be concentrated to a line, a circle or just be spread sparsely to cover a large region.

3 Optimization of Filter Coefficients

Consider for the moment that the number of filter components M and the coordinates for the nonzero coefficients for each filter are defined. The last step of the sequential filter optimization procedure is consequently to assign a value to each coefficient such that such that the difference between the reference function, $\check{F}(\mathbf{u})$, and $F(\mathbf{u})$ is minimized according to some distance measure. This distance is defined as the weighted square of the difference

$$D = \sum_{\Omega} W^2(\mathbf{u})||F(\mathbf{u}) - \check{F}(\mathbf{u})||^2 \qquad \Omega = \{\mathbf{u} : |u_i| < \pi\} \tag{4}$$

The purpose of the weighting function is to make the importance of a close fit proportional to the energy contribution. For most natural images the energy spectra decays as $W(\mathbf{u}) = |\mathbf{u}|^{-\alpha}$ where $1 < \alpha < 3$. In this paper $\alpha = 2$, for further discussion about the weighting function see [6].

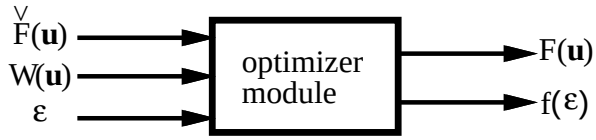

Figure 1: *The optimizer module have three inputs, the ideal filter function* $\check{F}(\mathbf{u})$*, the frequency weight* $W(\mathbf{u})$ *and the spatial coordinates* $\boldsymbol{\xi}$

The Optimizer Module

Consider for a moment the case where $M = 1$ in eq. (1). The coefficients of the convolution kernel that minimizes eq. (4) is computed by taking the partial derivative of D with respect to the N_1 kernel coefficients of $f_1(\boldsymbol{\xi})$. Setting the partial derivatives to zero results in a set of coupled equations from which the coefficients of $f_1(\boldsymbol{\xi})$ are computed. The computations are straightforward but cumbersome and are for that reason left out here. In fig. 1 such a single level filter optimizer is illustrated as a module having three inputs, the ideal filter function for the kernel $\check{F}(\mathbf{u})$, the weighting function $W(\mathbf{u})$ and the spatial coordinates for the nonzero coefficients $\boldsymbol{\xi}$. This optimization can unfortunately not be performed simultaneously for two or more filter components which is essential for the use of the sequential filter concept. It is, however, possible to use this method to optimize a single filter component with respect to the present value of the other $M - 1$ components. A recursive use of the filter optimization method can consequently be expected to converge rapidly if the coefficient coordinates of the M filter components enable a sufficient degree of freedom.

Sequential Filter Optimization

To extend the kernel optimizer for recursive use require some considerations as the ideal filter function and weight function for a certain filter component depends on the reference function $F(\mathbf{u})$ as well as the current value of the other $M - 1$ filter components. Consider the optimization of filter component $k \in [1, M]$. The ideal filter function $\check{F}(\mathbf{u})$ is expressed as

$$\check{F}_k(\mathbf{u}) = \frac{\check{F}(\mathbf{u})}{\prod\limits_{l \neq k} F_l(\mathbf{u})} \tag{5}$$

since in the ideal case

$$\check{F}(\mathbf{u}) = \prod_{k=1}^{M} F_k(\mathbf{u}) \tag{6}$$

according to eq. (1). For the weight function $W(\mathbf{u})$ the relation is not as obvious as for the ideal function but for reasons that will be apparent later on the weight

function for $\check{F}(\mathbf{u})$ is defined as

$$W_k(\mathbf{u}) = |\mathbf{u}|^{-2} \prod_{l \neq k} F_l(\mathbf{u}) \tag{7}$$

where $|\mathbf{u}|^{-2}$ relates to the energy spectra of the image. Insertion of eq. (5) and eq. (7) into the distance measure of eq. (4) results in

$$D = \sum_{\Omega} |\mathbf{u}|^{-4} (\prod_{l \neq k} F_l(\mathbf{u}))^2 [\frac{\check{F}(\mathbf{u})}{\prod_{l \neq k} F_l(\mathbf{u})} - F_k(\mathbf{u})]^2 \tag{8}$$

where $F_k(\mathbf{u})$ are defined in eq. (3). Simplifying eq. (8) yields:

$$D = \sum_{\Omega} |\mathbf{u}|^{-4} [\check{F}(\mathbf{u}) - \prod_{k=1}^{M} F_k(\mathbf{u})]^2 \tag{9}$$

which shows that the recursive optimization procedure will tend to minimize the difference between the combined effect of the component filters, $F_k(\mathbf{u})$, and the ideal filter $\check{F}(\mathbf{u})$.

A Recursive Algorithm for Sequential Filters

By using the result in eq. (5) and eq. (7) it is obvious how a set the optimizer modules can be connected to perform a recursive sequential filter optimization.The algorithm can be explained in the following steps

1. Set $F_2(\mathbf{u}), F_3(\mathbf{u}), \ldots, F_M(\mathbf{u}) = 1 \ \forall \mathbf{u}$
2. Define the nonzero coordinates for $f_1(\boldsymbol{\xi}), f_2(\boldsymbol{\xi}), \ldots f_M(\boldsymbol{\xi})$
3. For k=1 to M
   $$\text{opt}[f_k(\boldsymbol{\xi})] : \begin{cases} \check{F}_k(\mathbf{u}) & = & \check{F}(\mathbf{u}) / \prod_{l \neq k} F_l(\mathbf{u}) \\ W_k(\mathbf{u}) & = & |\mathbf{u}|^{-2} \prod_{l \neq k} F_l(\mathbf{u}) \end{cases}$$
4. Repeat step 3 until convergence (5-6 iterations)

Given enough degree of freedom in the spatial coordinates $\boldsymbol{\xi}$ the algorithm converges rapidly (typically 5-6 iterations). In the following parts of the paper this algorithm is applied to optimization of polar separable quadrature filters for 2D, 3D and 4D with M ranging from $2 - 4$.

4 Tensor Representation and Error Estimation

Before the actual filter implementation is presented it may be appropriate to discuss how the performance of the resulting filters are estimated. One obvious

possibility is to use the distance measure of eq. (4) to estimate the weighted distortion within the filter. Although such measurements are appropriate it is more important to estimate the performance of low-level vision tasks such as estimation of shape and orientation. To enable local orientation estimates the resulting quadrature filters are mapped to a *Local Tensor representation* [7],[6, ch 6]. Let q_k be the magnitudes of a set of polar separable quadrature filters implemented by the proposed sequential method. The local tensor $\mathbf{T}$ is attained by projecting the magnitude of the quadrature filter responses q_k onto a corresponding set of projection tensors $\mathbf{T_k}$

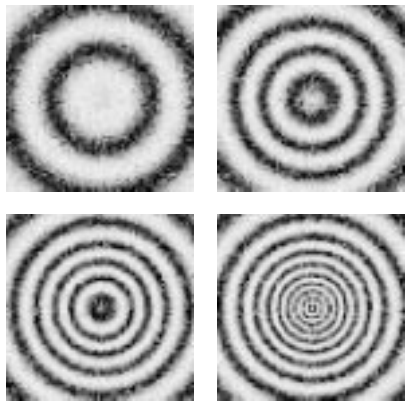

Figure 2: *Frame number* $5, 14, 21$ *and* 32 *of the* 64*-cube test sequence, SNR=10dB.*

$$\mathbf{T} = \sum_k \mathbf{T_k}\, q_k \tag{10}$$

where the $\mathbf{T_k}$ are symmetrical second order tensors of the same dimensionality as the signal. The computation of the projection tensor is based on the optimized sequential filter responses and not the ideal filter functions. This process enables consequently a compensation or whitening of the tensor metric due to any bias that are present in the filters or the filter orientations. A derivation of the *tensor whitening* method is found in [8]. The local tensor $\mathbf{T}$ constitute continuous geometrical representation of the neighbourhood. The distribution of the eigenvalues of $\mathbf{T}$ define the local shape of the neighbourhood. The most common case for natural images is the rank 1 case. Such a tensor can be written

$$\mathbf{T} \simeq \lambda_1 \hat{\mathbf{e}}_1 \hat{\mathbf{e}}_1^T \tag{11}$$

where $\hat{\mathbf{e}}_1$ is the eigenvector of $\mathbf{T}$ that corresponds to the largest eigenvalue λ_1. In 2D this corresponds to a local neighbourhood consisting of a line or an edge. In 3D, the environment is interpreted as a planar structure or (for time sequences) as a moving line. In 4D the rank 1 case corresponds to a moving plane. For further discussion on interpretation of the eigenvalues see

[6, ch 6]. The eigenvector $\hat{\mathbf{e}}_1$ corresponding to the largest eigenvalue define the orientation of the neighbourhood. To estimate the orientation error in the tensor representation require a test image containing rank 1 neighbourhoods and the ideal eigenvectors corresponding to this image. In 3D such a test image consists by a volume containing concentric spherical shells. Some frames from this test sequence are displayed in fig. 2. Locally all neighbourhoods are planar and all possible orientations are present. The angular RMS error $\Delta\varphi$ is finally computed as:

$$\Delta\varphi = \sin^{-1}\left(\sqrt{\frac{1}{2L}\sum_{l=1}^{L}\|\hat{x}\hat{x}^T - \hat{e_1}\hat{e_1}^T\|^2}\right) \tag{12}$$

where:

$\hat{x}$ is a unit vector in the correct orientation,
$\hat{e}_1$ is the eigenvector corresponding to the largest eigenvalue of the estimated tensor T,
L is the number of points and
$\Delta\varphi$ is the angular RMS error.

5 A Filtering Structure for 2D Signals

As mentioned earlier, the examples in this paper are focused on polar separable quadrature filters. The reason for this is the simplicity by which a robust and continuous tensor representation can be attained from this type of filters. This does not imply that this filtering approach is limited to this types of filters. In common for 2D, 3D and the 4D case is that the radial part of $\check{F}(\mathbf{u})$ is of bandpass type with a center frequency of $\pi/(2\sqrt{2}) \approx 1.11$ and a bandwidth of 2 octaves. The orientation of each quadrature filter is defined by a vector $\hat{\mathbf{n}}$ and the angular function of $\check{F}(\mathbf{u})$ is defined as $(\hat{\mathbf{u}} \cdot \hat{\mathbf{n}})^2$ for $(\mathbf{u} \cdot \hat{\mathbf{n}}) > 0$ and zero elsewhere. The number of degrees of freedom that are present in a symmetric second order tensor is

$$\frac{(d+1)d}{2} \tag{13}$$

where d is the dimensionality of the signal. In two dimensions the minimal number of filters is consequently three. Using sequential filters where each filter component only contain few coefficients it is of out most importance to make advantage of all present symmetry aspects. The filter orienting vectors, $\hat{\mathbf{n}}$, must consequently be chosen with care using sequential filters as opposed to conventional FIR-filtering. The most efficient choice is to direct the filters along the main coordinate axes which takes care of two filters. The second best alternative is to orient the filters symmetrically between the main coordinate axes which also gives two alternatives. From earlier experiences on conventional filters we

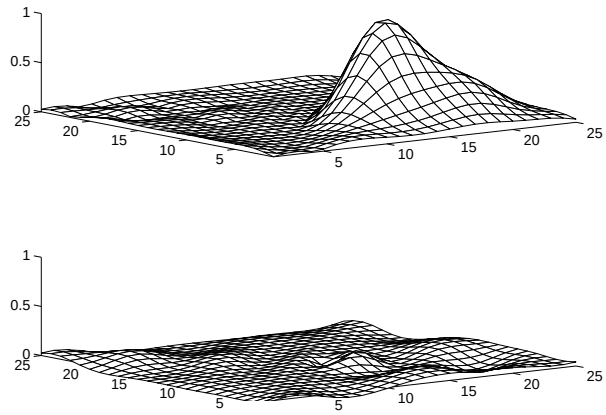

Figure 3: *Upper part: Resulting Filter function in the FD. Lower part: Deviation from the ideal filter function.*

know that using four filters increase the robustness in noisy and ambiguous neighbourhoods considerably in 2D. The the filter orienting vectors for 2D are therefore defined as:

$$\begin{aligned}
\hat{\mathbf{n}}_1 &= \quad\quad\quad (\ 1, \quad 0\ )^T \\
\hat{\mathbf{n}}_2 &= 1/\sqrt{2}\ (\ 1, \quad 1\ )^T \\
\hat{\mathbf{n}}_3 &= \quad\quad\quad (\ 0, \quad 1\ )^T \\
\hat{\mathbf{n}}_4 &= 1/\sqrt{2}\ (-1, \quad 1\ )^T
\end{aligned} \tag{14}$$

The following problem is to decide how to separate the quadrature filters in the above directions into sequential filter set. Since the filters are two dimensional it seems reasonable that (M=2) filter components should be sufficient. The choice of the coordinates is dependent of the type of filter that is to be implemented but in this case it is probably most efficient to concentrate the coefficients along a line. Again with the symmetry aspect in mind the nonzero coefficient of these two filters are distributed along with, and orthogonal to the filter orienting vectors in eq. (14). The sequential filtering structure for sequential filters in 2D is for this approach expressed as

$$\begin{aligned}
\mathbf{q}_1(\boldsymbol{\xi}) &= \quad f_y(\boldsymbol{\xi}) * \quad \mathbf{f}_x(\boldsymbol{\xi}) \\
\mathbf{q}_2(\boldsymbol{\xi}) &= f_{-x,y}(\boldsymbol{\xi}) * \quad \mathbf{f}_{x,y}(\boldsymbol{\xi}) \\
\mathbf{q}_3(\boldsymbol{\xi}) &= \quad f_x(\boldsymbol{\xi}) * \quad \mathbf{f}_y(\boldsymbol{\xi}) \\
\mathbf{q}_4(\boldsymbol{\xi}) &= \quad f_{x,y}(\boldsymbol{\xi}) * \mathbf{f}_{-x,y}(\boldsymbol{\xi})
\end{aligned} \tag{15}$$

where the indices indicate the distribution of the nonzero coefficients of the filter components. The filtering is performed in two steps, the first filter component is

	M=2 (132 coeff)		M=1 (648 coeff)	
SNR	Ang. err	Density(%)	Ang. err	Density(%)
0dB	15.99°	59.7	15.73°	60.7
2dB	9.54°	71.7	9.39°	72.8
4dB	5.79°	81.6	5.71°	82.5
6dB	3.66°	88.1	3.62°	88.9
8dB	2.42°	91.9	2.39°	92.8
10dB	1.69°	93.9	1.66°	95.1
15dB	1.01°	95.6	0.99°	97.3
∞	0.90°	95.9	0.88°	97.7

Table 1: *Performance of the sequential 2D filters in comparison to a direct implementation.*

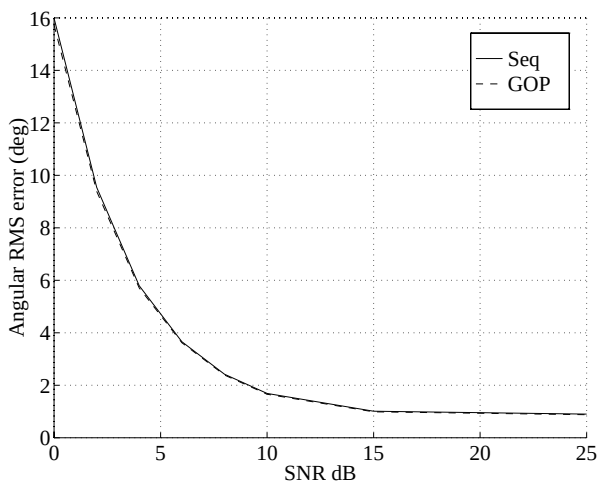

Figure 4: *Angular RMS error for different SNR. The solid line refers to the sequential (M=2) implementation while the dashed line corresponds to a traditional (M=1) implementation*

real valid and is expected to be of low pass type. The second filter component, which is directed in the main direction of the quadrature filter, is expected to be complex valued.

The resulting optimized sequential filter set are found below together with the corresponding projection tensors. The spatial extension of the filter components vary from 5-9 pixels. Note that only four filters need to be computed. The remaining filters are obtained by mirroring according to the indices in eq. (15). The reason for $\mathbf{f}_{-x,y}(\boldsymbol{\xi})$ and $\mathbf{f}_{-x,y}(\boldsymbol{\xi})$ to be tri-diagonal is simply to obtain sufficient sampling density.

The resulting frequency response for $\mathbf{q}_2(\boldsymbol{\xi})$ in eq. (15) is displayed in the

upper part of fig. 3. The lower part of the same figure show the deviation from the ideal filter $\check{F}(\mathbf{u})$. The weighted distance measure of eq. (4) accumulates to $D = 0.22$. In comparison to a traditional implementation using a single complex filter of size 9×9 the distance measure is twice as high. The weighted distance measure is, however, of limited use when the performance of a filter set is to be judged. To obtain a more relevant comparison the local tensor $\mathbf{T}$ is computed from these filter responses using the projection tensors below. This computation where performed for both the sequential filters and the traditional ($M = 1$) implementation for different SNR. From the tensor data the angular RMS error was computed according to eq. (12). The result the result is listed in table (1) and exceeds our most optimistic expectations as there are no detectable loss in accuracy while the total number of coefficients used in the sequential implementation is 132 as opposed to 648 for a direct implementation. The computational complexity is consequently reduced **5 times** in this example. The density in table (1) is a graded coverage estimate for the tensor representation. The density defines to how well the local one dimensional neighbourhoods of the test image is reflected in the tensor representation. In contrast to the angular RMS error the density only refers to the shape of the neighbourhood and is computed from the two largest eigen values λ_1 and λ_2 of the local tensor $\mathbf{T}$.

$$\text{Density} = \frac{\sum \left(\frac{\lambda_1 - \lambda 2}{\lambda 1}\right)^2 \lambda_1^2}{\sum \lambda_1^2} \tag{16}$$

The difference in density between the sequential and the direct method is in the same magnitude as for the angular RMS error, i.e. insignificant. From the measurements in table (1) it is concluded that the sequential filtering approach support an efficient and robust tensor representation in 2D.

6 A Filtering Structure for 3D Signals

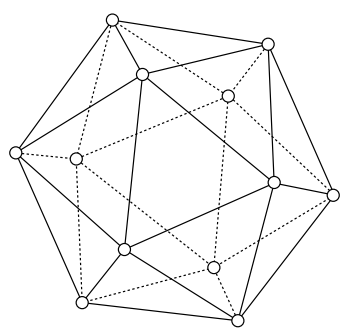

Figure 5: *The icosahedron, one of 5 platonic polyhedra.*

In 3D, the minimal number of filters required to produce a second order tensor representation is six, eq. (13). The apparent choice using conventional filter-

ing is to define the filter orienting vectors as the vertices of a hemi-icosahedron [7, 6]. The icosahedron is a regular diametrically symmetric polyhedron (fig. 5) which implies that the filters are equally spread. The six filter orienting vectors defined by the icosahedron are

$$\begin{array}{llll} \hat{\mathbf{n}}_1 = c\,(\ a\ ,\ 0\ ,\ b\ )^T & \hat{\mathbf{n}}_2 = c\,(-a\ ,\ 0\ ,\ b\ )^T \\ \hat{\mathbf{n}}_3 = c\,(\ b\ ,\ a\ ,\ 0\ )^T & \hat{\mathbf{n}}_4 = c\,(\ b\ ,-a\ ,\ 0\ )^T \\ \hat{\mathbf{n}}_5 = c\,(\ 0\ ,\ b\ ,\ a\ )^T & \hat{\mathbf{n}}_6 = c\,(\ 0\ ,\ b\ ,-a\ )^T \end{array} \tag{17}$$

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

These orientations are unfortunately not suitable for a sequential implementation because of the symmetry aspects discussed earlier. In agreement with the 2D case the first choice of filter orienting vectors for sequential filtering in 3D are along the coordinate axes (3 possibilities) and the second symmetrically between two coordinate axes (6 possibilities). Although 6 filters are sufficient in theory we choose to use nine filters for two reasons. The first reason is that the orientation error due to the sequential filters not being polar separable is reduced by increasing the number of filters. The second reason is that using the convolver tree structure presented below, the increase in computational complexity by computing 9 filter responses instead of 6 is very small. The 9 filter orienting vectors for 3D are consequently defined as

$$\begin{array}{lrrr} \hat{\mathbf{n}}_1 = 1/\sqrt{2}\ (\ 0, & -1, & 1\ )^T \\ \hat{\mathbf{n}}_2 = 1/\sqrt{2}\ (\ 0, & 1, & 1\ )^T \\ \hat{\mathbf{n}}_3 = \quad (\ 0, & 1, & 0\ )^T \\ \hat{\mathbf{n}}_4 = 1/\sqrt{2}\ (-1, & 0, & 1\ )^T \\ \hat{\mathbf{n}}_5 = 1/\sqrt{2}\ (\ 1, & 0, & 1\ )^T \\ \hat{\mathbf{n}}_6 = \quad (\ 0, & 0, & 1\ )^T \\ \hat{\mathbf{n}}_7 = 1/\sqrt{2}\ (\ 1, & 1, & 0\ )^T \\ \hat{\mathbf{n}}_8 = 1/\sqrt{2}\ (-1, & 1, & 0\ )^T \\ \hat{\mathbf{n}}_9 = \quad (\ 1, & 0, & 0\ )^T \end{array}$$

These filter orientations are not exactly equally spread as in the filters orientations in eq. (17). To eliminate any orientation bias in the local tensor $\mathbf{T}$ the projection tensors are computed using the *tensor whitening method* discussed earlier, [8]. The sequential filtering structure for the 3D case is consequently

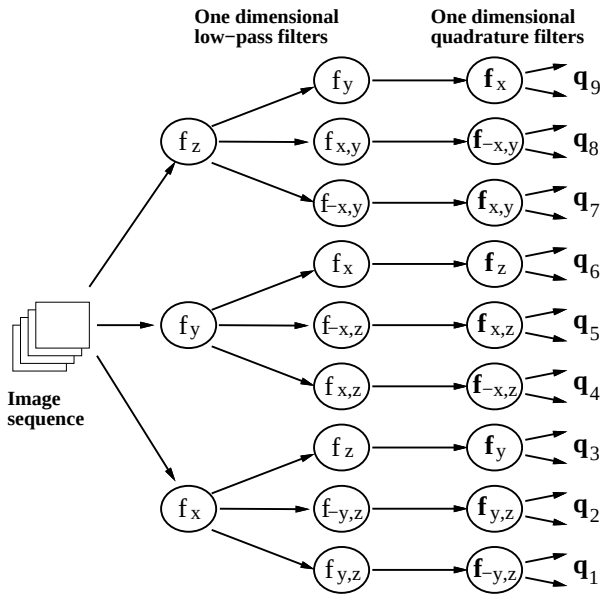

Figure 6: *21 convolvers organized in 3 levels. Only 3 filters are required on the first level.*

expressed as

$$
\begin{aligned}
\mathbf{q}_1(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * \quad f_{y,z}(\boldsymbol{\xi}) * \mathbf{f}_{-y,z}(\boldsymbol{\xi}) \\
\mathbf{q}_2(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * f_{-y,z}(\boldsymbol{\xi}) * \quad \mathbf{f}_{y,z}(\boldsymbol{\xi}) \\
\mathbf{q}_3(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * \quad f_z(\boldsymbol{\xi}) * \quad \mathbf{f}_y(\boldsymbol{\xi}) \\
\mathbf{q}_4(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * \quad f_{x,z}(\boldsymbol{\xi}) * \mathbf{f}_{-x,z}(\boldsymbol{\xi}) \\
\mathbf{q}_5(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * f_{-x,z}(\boldsymbol{\xi}) * \quad \mathbf{f}_{x,z}(\boldsymbol{\xi}) \\
\mathbf{q}_6(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * \quad f_x(\boldsymbol{\xi}) * \quad \mathbf{f}_z(\boldsymbol{\xi}) \\
\mathbf{q}_7(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * f_{-x,y}(\boldsymbol{\xi}) * \quad \mathbf{f}_{x,y}(\boldsymbol{\xi}) \\
\mathbf{q}_8(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * \quad f_{x,y}(\boldsymbol{\xi}) * \mathbf{f}_{-x,y}(\boldsymbol{\xi}) \\
\mathbf{q}_9(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * \quad f_y(\boldsymbol{\xi}) * \quad \mathbf{f}_x(\boldsymbol{\xi})
\end{aligned}
\tag{18}
$$

Each quadrature filter response are computed in three steps. The first two components are real valued and are both oriented in directions orthogonal to the filter orienting vectors. The last component is complex valued and directed along the main direction of the ideal filter. In the sequential filter organization above, the initial filtering step is identical for several filters. This observation is used to define an efficient convolver tree structure of fig. 6 consisting of 21 convolvers on 3 levels. Using only 3 convolvers at the top level. The optimized sequential filter components are displayed below. In fig. 7 the resulting frequency response for $\mathbf{q}_7(\boldsymbol{\xi})$ is illustrated in the $u1, u2$-plane. The angular RMS error of the local tensor were computed for different SNR and the result is given in table 2. The

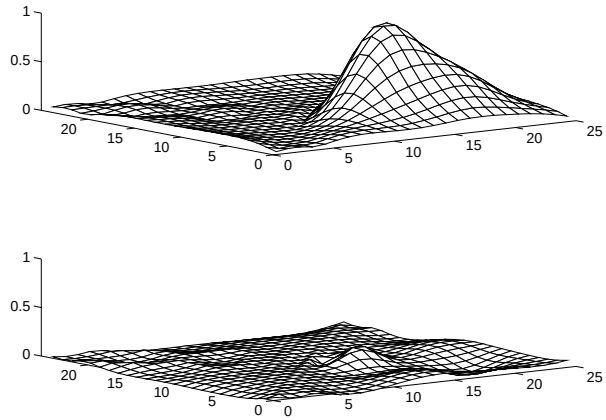

Figure 7: *Upper part: Frequency response for* $\mathbf{q}_7(\boldsymbol{\xi})$ *in the* u_1, u_2*-plane. Lower part: Deviation from the ideal filter function.*

left column correspond to a filter set consisting of 9 quadrature filters where each quadrature filter response are computed sequentially by three filter components according to fig. 6 The computation of *nine quadrature filter responses requires* 345 *multiplications.* If on the other hand conventional quadrature filters are used, *six complex filters of size* $9 \times 9 \times 9$ *require almost* 9000 *multiplications.* The computational load is consequently *reduced* 25 *times* in this example while the difference in accuracy is insignificant, see table 2.

SNR	$M = 3$ 345 coeff.	$M = 1$ 8748 coeff.
∞	$0.76°$	$0.71°$
10dB	$3.02°$	$3.03°$
0dB	$9.35°$	$9.69°$

Table 2: *Angular RMS error in degrees for a sequential filter with* 3 *components and the corresponding filter implemented by a single component.*

7 A Filtering Structure for 4D Signals

In 4D, the choice of filter orienting vectors is quite simple. From eq. (13) we conclude the the the minimum number of filters required to produce a tensor representation in 4D is 10. If the filters are localized symmetrically between

two main coordinate axes this results in 12 possibilities, which is sufficient. The filter orienting vectors in 4D are defined as

$$
\begin{aligned}
\hat{\mathbf{n}}_1 &= 1/\sqrt{2}\ (\ 0, \quad 0, \quad 1, \quad 1\ )^T \\
\hat{\mathbf{n}}_2 &= 1/\sqrt{2}\ (\ 0, \quad 0, \quad 1, \quad -1\ )^T \\
\hat{\mathbf{n}}_3 &= 1/\sqrt{2}\ (\ 0, \quad 1, \quad 1, \quad 0\ )^T \\
\hat{\mathbf{n}}_4 &= 1/\sqrt{2}\ (\ 0, \quad -1, \quad 1, \quad 0\ )^T \\
\hat{\mathbf{n}}_5 &= 1/\sqrt{2}\ (\ 1, \quad 0, \quad 0, \quad 1\ )^T \\
\hat{\mathbf{n}}_6 &= 1/\sqrt{2}\ (\ 1, \quad 0, \quad 0, \quad -1\ )^T \\
\hat{\mathbf{n}}_7 &= 1/\sqrt{2}\ (\ 1, \quad 0, \quad 1, \quad 0\ )^T \\
\hat{\mathbf{n}}_8 &= 1/\sqrt{2}\ (-1, \quad 0, \quad 1, \quad 0\ )^T \\
\hat{\mathbf{n}}_9 &= 1/\sqrt{2}\ (\ 0, \quad 1, \quad 0, \quad 1\ )^T \\
\hat{\mathbf{n}}_{10} &= 1/\sqrt{2}\ (\ 0, \quad 1, \quad 0, \quad -1\ )^T \\
\hat{\mathbf{n}}_{11} &= 1/\sqrt{2}\ (\ 1, \quad 1, \quad 0, \quad 0\ )^T \\
\hat{\mathbf{n}}_{12} &= 1/\sqrt{2}\ (-1, \quad 1, \quad 0, \quad 0\ )^T
\end{aligned}
\tag{19}
$$

These 12 filter orientations also correspond to the vertices of the 24-cell, one of three regular polyhedra in 4D. The above filter orientations are consequently equally spread in the 4D signal space which simplifies to computation of the projection tensors.

In agreement with the 3D case the sequential filtering structure for 4D signals is expressed as

$$
\begin{aligned}
\mathbf{q}_1(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * f_y(\boldsymbol{\xi}) * f_{-z,w}(\boldsymbol{\xi}) * \mathbf{f}_{z,w}(\boldsymbol{\xi}) \\
\mathbf{q}_2(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * f_y(\boldsymbol{\xi}) * f_{z,w}(\boldsymbol{\xi}) * \mathbf{f}_{z,-w}(\boldsymbol{\xi}) \\
\mathbf{q}_3(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{-y,z}(\boldsymbol{\xi}) * \mathbf{f}_{y,z}(\boldsymbol{\xi}) \\
\mathbf{q}_4(\boldsymbol{\xi}) &= f_x(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{y,z}(\boldsymbol{\xi}) * \mathbf{f}_{-y,z}(\boldsymbol{\xi}) \\
\mathbf{q}_5(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * f_z(\boldsymbol{\xi}) * f_{-x,w}(\boldsymbol{\xi}) * \mathbf{f}_{x,w}(\boldsymbol{\xi}) \\
\mathbf{q}_6(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * f_z(\boldsymbol{\xi}) * f_{x,w}(\boldsymbol{\xi}) * \mathbf{f}_{x,-w}(\boldsymbol{\xi}) \\
\mathbf{q}_7(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{-x,z}(\boldsymbol{\xi}) * \mathbf{f}_{x,z}(\boldsymbol{\xi}) \\
\mathbf{q}_8(\boldsymbol{\xi}) &= f_y(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{x,z}(\boldsymbol{\xi}) * \mathbf{f}_{-x,z}(\boldsymbol{\xi}) \\
\mathbf{q}_9(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * f_x(\boldsymbol{\xi}) * f_{-y,w}(\boldsymbol{\xi}) * \mathbf{f}_{y,w}(\boldsymbol{\xi}) \\
\mathbf{q}_{10}(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * f_x(\boldsymbol{\xi}) * f_{y,w}(\boldsymbol{\xi}) * \mathbf{f}_{y,-w}(\boldsymbol{\xi}) \\
\mathbf{q}_{11}(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{-x,y}(\boldsymbol{\xi}) * \mathbf{f}_{x,y}(\boldsymbol{\xi}) \\
\mathbf{q}_{12}(\boldsymbol{\xi}) &= f_z(\boldsymbol{\xi}) * f_w(\boldsymbol{\xi}) * f_{x,y}(\boldsymbol{\xi}) * \mathbf{f}_{-x,y}(\boldsymbol{\xi})
\end{aligned}
$$

The filtering is performed in 4 levels where the three first are real valued and the last filter is complex. The distribution of the nonzero coefficients within each filter component is indicated by the indices. Note that the indices of the

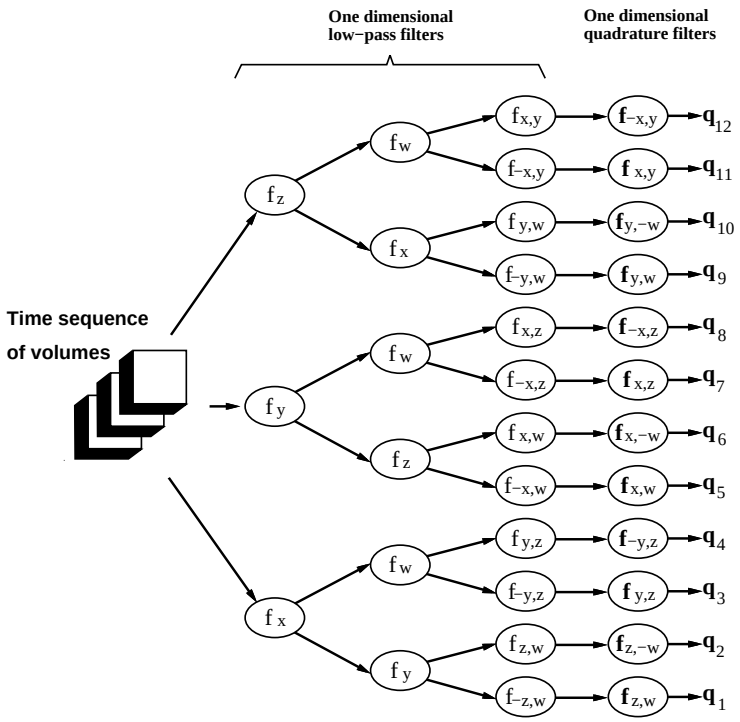

Figure 8: *Convolver organization in 4D, 33 convolvers in 4 levels*

components within each filter are orthogonal and that in the two first filtering steps the same filter combination occurs several times. I fig. 8 a 4D convolver structure in 4 levels is displayed. By careful combination of the results in the sequential filtering structure only 3 and 6 convolvers are required at the first and second level respectively. The optimized filters with center frequency $\varrho_0 = 1.11$ and bandwidth $B = 2$ octaves are found below. At this point it starts to get really interesting to compare the the computational complexity of sequential convolution to traditional implementations. The spatial extent of the optimized filter components vary from 5-9 pixels. To compute a single quadrature filter response require $7+7+5+19+18 = 56$ real multiplications. Using the convolver tree structure of fig. 8 reduces this figure further by about 15 % resulting in 567 multiplications for 12 filters. In conventional FIR-filtering a complex filter of size $9 \times 9 \times 9 \times 9$ require almost *160 000 real valued multiplication to compute 12 quadrature filter responses.* The sequential approach is consequently *almost 280 times more efficient.*

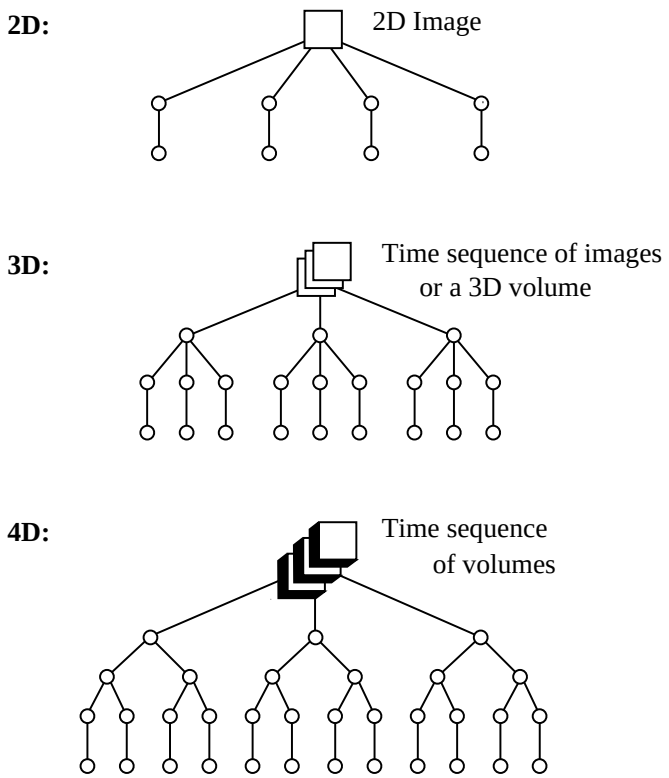

Figure 9: *Sequential convolver organization in 2D,3D and 4D*

8 Implementation Aspects

The general convolver module is implemented in C with the graphical user interface in AVS (Application Visualization System), see [10] for more details. The tree structure of the convolver organization, fig. 9, provide an efficient optimization of the processing performance for a wide variety computer architectures. The soft implementation of the convolver network enable a simple reconfiguration for different types of input data (different outer dimensions) e.g 2D image, 3D spatiotemporal image sequences (3D volumes) and 4D volume sequences. The vector length (i.e. the inner dimension) of the kernel data is likewise simple to adapt from e.g scalars, RGB-signals, 2D vector fields, 3D vector fields, tensor fields etc.

9 Acknowledgment

The support of the National Swedish Board for Technical Development, NUTEK (project 9305120-1) is greatfully acknowledged.

A Sequential filters for 2D

These are the sequential filter components used in the experiments. From these four filter components all filters in eq. (15) can be produced by mirroring according to the indices. The coefficients are converted to integers to support an efficient implementation.

$$f_x(\boldsymbol{\xi}) = \begin{bmatrix} 1256 & 4195 & 14278 & 24981 & 14278 & 4195 & 1256 \end{bmatrix}$$

$$f_{xy}(\boldsymbol{\xi}) = \begin{bmatrix} & & & & & & 613 \\ & & & & & 8593 & \\ & & & & 24766 & & \\ & & & 8593 & & & \\ & & 613 & & & & \end{bmatrix}$$

$$Re[\mathbf{f}_x(\boldsymbol{\xi})] = \begin{bmatrix} -287 & -673 & -1507 & 92 & 4748 & 92 & -1507 & -673 & -287 \end{bmatrix}$$

$$Im[\mathbf{f}_x(\boldsymbol{\xi})] = \begin{bmatrix} 607 & 5 & -405 & -3683 & 0 & 3683 & 405 & -5 & -607 \end{bmatrix}$$

$$Re[\mathbf{f}_{xy}] = \begin{bmatrix} & & & & & -246 & -187 \\ & & & & -689 & -869 & -246 \\ & & & 932 & -1454 & -689 & \\ & & 932 & 5035 & 932 & & \\ & -689 & -1454 & 932 & & & \\ -246 & -869 & -689 & & & & \\ -187 & -246 & & & & & \end{bmatrix}$$

$$Im[\mathbf{f}_{xy}] = \begin{bmatrix} & & & & & -219 & -484 \\ & & & & 273 & -139 & -219 \\ & & & 2144 & 2101 & 273 & \\ & & -2144 & & 2144 & & \\ & -273 & -2101 & -2144 & & & \\ 219 & 139 & -273 & & & & \\ 484 & 219 & & & & & \end{bmatrix}$$

Projection tensors in 2D

$$\mathbf{T}_1 = \begin{pmatrix} 0.75 & 0.00 \\ 0.00 & -0.25 \end{pmatrix} \qquad \mathbf{T}_2 = \begin{pmatrix} 0.25 & 0.50 \\ 0.50 & 0.25 \end{pmatrix}$$

$$\mathbf{T}_3 = \begin{pmatrix} -0.25 & 0.00 \\ 0.00 & 0.75 \end{pmatrix} \qquad \mathbf{T}_4 = \begin{pmatrix} 0.25 & -0.50 \\ -0.50 & 0.25 \end{pmatrix}$$

B Sequential Filters for 3D

$$f_x(\boldsymbol{\xi}) = \begin{bmatrix} 1277 & 4060 & 15481 & 28833 & 15481 & 4060 & 1277 \end{bmatrix}$$

$$f_{xy}(\boldsymbol{\xi}) = \begin{bmatrix} & & & & & & 24 \\ & & & & & 492 & \\ & & & & 1565 & & \\ & & & 492 & & & \\ & & 24 & & & & \end{bmatrix}$$

$$Re[\mathbf{f}_x(\boldsymbol{\xi})] = \begin{bmatrix} -121 & -237 & -612 & -51 & 2041 & -51 & -612 & -237 & -121 \end{bmatrix}$$

$$Im[\mathbf{f}_x(\boldsymbol{\xi})] = \begin{bmatrix} 206 & -28 & -75 & -1569 & 0 & 1569 & 75 & 28 & -206 \end{bmatrix}$$

$$Re[\mathbf{f}_{xy}] = \begin{bmatrix} & & & & & & -1641 & -1538 \\ & & & & & -4817 & -5819 & -1641 \\ & & & & 6454 & -12369 & -4817 & \\ & & & 6454 & 39469 & 6454 & & \\ & & -4817 & -12369 & 6454 & & & \\ & -1641 & -5819 & -4817 & & & & \\ & -1538 & -1641 & & & & & \end{bmatrix}$$

$$Im[\mathbf{f}_{xy}] = \begin{bmatrix} & & & & & & -1297 & -2447 \\ & & & & & 1495 & -1106 & -1297 \\ & & & & 17177 & 14544 & 1495 & \\ & & & -17177 & & 17177 & & \\ & & -1495 & -14544 & -17177 & & & \\ & 1297 & 1106 & -1495 & & & & \\ & 2447 & 1297 & & & & & \end{bmatrix}$$

Projection Tensors in 3D

These projection tensors where computed using the tensor whitening method ([8]) to compensate for any orientation bias due to uneven spread of the filters.

$$\mathbf{T}_1 = \begin{pmatrix} 0.2690 & 0 & 0 \\ 0 & 1.0042 & 1.5313 \\ 0 & 1.5313 & 1.0042 \end{pmatrix} \qquad \mathbf{T}_2 = \begin{pmatrix} 0.2690 & 0 & 0 \\ 0 & 1.0042 & -1.5313 \\ 0 & -1.5313 & 1.0042 \end{pmatrix}$$

$$\mathbf{T}_3 = \begin{pmatrix} -0.1804 & 0 & 0 \\ 0 & 2.2538 & 0 \\ 0 & 0 & -0.1804 \end{pmatrix} \qquad \mathbf{T}_4 = \begin{pmatrix} 1.0042 & 0 & -1.5313 \\ 0 & 0.2690 & 0 \\ -1.5313 & 0 & 1.0042 \end{pmatrix}$$

$$\mathbf{T}_5 = \begin{pmatrix} 1.0042 & 0 & 1.5313 \\ 0 & 0.2690 & 0 \\ 1.5313 & 0 & 1.0042 \end{pmatrix} \qquad \mathbf{T}_6 = \begin{pmatrix} -0.1804 & 0 & 0 \\ 0 & -0.1804 & 0 \\ 0 & 0 & 2.2538 \end{pmatrix}$$

$$\mathbf{T}_7 = \begin{pmatrix} 1.0042 & -1.5313 & 0 \\ -1.5313 & 1.0042 & 0 \\ 0 & 0 & 0.2690 \end{pmatrix} \qquad \mathbf{T}_8 = \begin{pmatrix} 1.0042 & 1.5313 & 0 \\ 1.5313 & 1.0042 & 0 \\ 0 & 0 & 0.2690 \end{pmatrix}$$

$$\mathbf{T}_9 = \begin{pmatrix} 2.2538 & 0 & 0 \\ 0 & -0.1804 & 0 \\ 0 & 0 & -0.1804 \end{pmatrix}$$

C Sequential Filters for 4D

$$f_x(\boldsymbol{\xi}) = \begin{bmatrix} 201 & 1891 & 14815 & 31267 & 14815 & 1891 & 201 \end{bmatrix}$$

$$f_{xy}(\boldsymbol{\xi}) = \begin{bmatrix} -963 & & & & & \\ & 6837 & & & & \\ & & 28702 & & & \\ & & & 6837 & & \\ & & & & -963 & \end{bmatrix}$$

$$Re[\mathbf{f}_{xy}] = \begin{bmatrix} & & & & & 38 & 345 \\ & & & & -2980 & -1804 & 38 \\ & & & 2811 & -11172 & -2980 & \\ & & 2811 & 25786 & 2811 & & \\ & -2980 & -11172 & 2811 & & & \\ 38 & -1804 & -2980 & & & & \\ 345 & 38 & & & & & \end{bmatrix}$$

$$
Im[\mathbf{f}_{xy}] = \begin{bmatrix}
 & & & & & -963 & -785 \\
 & & & & -1310 & -2221 & -963 \\
 & & & 12279 & 7172 & -1310 & \\
 & & -12279 & & 12279 & & \\
 & 1310 & -7172 & -12279 & & & \\
963 & 2221 & 1310 & & & & \\
785 & 963 & & & & &
\end{bmatrix}
$$

Projection Tensors in 4D

$$
\mathbf{T}_1 = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & -0.50 & 0 & 0 \\ 0 & 0 & 1.00 & 1.50 \\ 0 & 0 & 1.50 & 1.00 \end{pmatrix}
\qquad
\mathbf{T}_2 = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & -0.50 & 0 & 0 \\ 0 & 0 & 1.00 & -1.50 \\ 0 & 0 & -1.50 & 1.00 \end{pmatrix}
$$

$$
\mathbf{T}_3 = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & 1.00 & 1.50 & 0 \\ 0 & 1.50 & 1.00 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
\qquad
\mathbf{T}_4 = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & 1.00 & -1.50 & 0 \\ 0 & -1.50 & 1.00 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
$$

$$
\mathbf{T}_5 = \begin{pmatrix} 1.00 & 0 & 0 & 1.50 \\ 0 & -0.50 & 0 & 0 \\ 0 & 0 & -0.50 & 0 \\ 1.50 & 0 & 0 & 1.00 \end{pmatrix}
\qquad
\mathbf{T}_6 = \begin{pmatrix} 1.00 & 0 & 0 & -1.50 \\ 0 & -0.50 & 0 & 0 \\ 0 & 0 & -0.50 & 0 \\ -1.50 & 0 & 0 & 1.00 \end{pmatrix}
$$

$$
\mathbf{T}_7 = \begin{pmatrix} 1.00 & 0 & 1.50 & 0 \\ 0 & -0.50 & 0 & 0 \\ 1.50 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
\qquad
\mathbf{T}_8 = \begin{pmatrix} 1.00 & 0 & -1.50 & 0 \\ 0 & -0.50 & 0 & 0 \\ -1.50 & 0 & 1.00 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
$$

$$
\mathbf{T}_9 = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & 1.00 & 0 & 1.50 \\ 0 & 0 & -0.50 & 0 \\ 0 & 1.50 & 0 & 1.00 \end{pmatrix}
\qquad
\mathbf{T}_{10} = \begin{pmatrix} -0.50 & 0 & 0 & 0 \\ 0 & 1.00 & 0 & -1.50 \\ 0 & 0 & -0.50 & 0 \\ 0 & -1.50 & 0 & 1.00 \end{pmatrix}
$$

$$
\mathbf{T}_{11} = \begin{pmatrix} 1.00 & 1.50 & 0 & 0 \\ 1.50 & 1.00 & 0 & 0 \\ 0 & 0 & -0.50 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
\qquad
\mathbf{T}_{12} = \begin{pmatrix} 1.00 & -1.50 & 0 & 0 \\ -1.50 & 1.00 & 0 & 0 \\ 0 & 0 & -0.50 & 0 \\ 0 & 0 & 0 & -0.50 \end{pmatrix}
$$

References

[1] M. Andersson. *Controllable Multidimensional Filters in Low Level Computer Vision.* PhD thesis, Linköping University, Sweden, S–581 83 Linköping, Sweden, September 1992. Dissertation No 282, ISBN 91–7870–981–4.

[2] M. T. Andersson and H. Knutsson. Controllable 3-D Filters for Low Level Computer Vision. In *Proceedings of the 8th Scandinavian Conference on Image Analysis*, Tromsø, May 1993. SCIA.

[3] R. Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, 2nd edition, 1986.

[4] D. E. Dudgeon and R. M. Mersereau. *Multidimensional Digital Signal Processing.* Prentice-Hall signal processing series. Prentice-Hall, 1984. ISBN 0-13-604959-1.

[5] G. H. Granlund. In search of a general picture processing operator. *Computer Graphics and Image Processing*, 8(2):155–178, 1978.

[6] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[7] H. Knutsson. Representing local structure using tensors. In *The 6th Scandinavian Conference on Image Analysis*, pages 244–251, Oulu, Finland, June 1989. Report LiTH–ISY–I–1019, Computer Vision Laboratory, Linköping University, Sweden, 1989.

[8] H. Knutsson and M. Andersson. Robust N-Dimensional Orientation Estimation using Quadrature Filters and Tensor Whitening. In *Proceedings of IEEE International Conference on Acoustics, Speech, & Signal Processing*, Adelaide, Australia, April 1994. IEEE. LiTH-ISY-R-1798.

[9] H. Knutsson, M. Andersson, and J Wiklund. Advanced Filter Design. In *Proceedings of the Scandinavian Conference on Image analysis*, June 1999. Submitted.

[10] J. Wiklund and H. Knutsson. A Generalized Convolver. In *Proceedings of the 9th Scandinavian Conference on Image Analysis*, Uppsala, Sweden, June 1995. SCIA.