A 3D Isotropic Quadrature Filter for Motion Estimation Problems*

Martin Krause and Gerald Sommer

Institute of Computer Science and Applied Mathematics

University of Kiel, Germany

ABSTRACT

Quadrature filters[5] are a well known means for local spectral analysis of images and to extract relevant structure. Recently[9], there has been the discovering of an isotropic quadrature filter for images that does not need steering with respect to orientation and provides the user with geometric (local orientation) and structural (local phase) information. Here, we present a further extension of this, an *isotropic quadrature filter for 3D data.* With only four convolutions we can calculate the local orientation, energy and the phase of locally intrinsically 1D structures[†] in 3D image data. There is a wide variety of useful applications for a filter of this kind. In this paper, we will restrict ourselves to the application of phase-based motion or flow field estimation.

1. INTRODUCTION

When we want to do *optical flow field estimation*, there are several approaches to calculate the well known *motion constraint.*

- *Gradient-based* methods use the gray-value gradient of two different images to obtain a three dimensional vector that is perpendicular to the pixel motion between the two frames. A major drawback of this method is that it is not very robust with respect to gray-value changes – as introduced by changing lighting or just noise – from one image to the next.
- *Phase-based* methods use quadrature filters to calculate a 2D *phase vector field* for each of the two frames to compare. The local orientation of the phase vector is given by the Gabor filter with the highest local energy and the magnitude of it by the calculated phase itself. Then the Gabor phase derivative with respect to the time axis is calculated and added as third component to the phase vector field to archive a three dimensional motion constraint field. The advantage compared with gradient-based methods is clearly that phase, as a description of structure, is not as susceptible to lighting changes as gray-value gradients. One drawback of this method is that we have to handle phase wrappings while calculating the temporal derivative. Another disadvantage is that we have to sample the orientation of structures by steering the filters.

We focus on the phase-based methods and propose some improvements to overcome the disadvantages mentioned.

This paper deals with the generalisation of the *Hilbert transform*[13, 15] and the closely related *analytic signal* to higher dimensions. We extend the thesis of Felsberg[9] – a generalisation of the Hilbert transform to 2D – to 3D images. So, if one wants to get the full picture, the reading of his awarded thesis is highly recommended. Both, Felsberg[9] and this paper, as well as its more elaborate form[18], make frequent use of *geometric algebra.* We recommend the tutorials[7, 8, 14, 19] to get an idea of it or to read the first chapters of the book of its founder

*This research was granted by EC contract IST-2001-3422 (VISATEC project)

Further author information: (Send correspondence to M. Krause)
M. Krause: mk@ks.informatik.uni-kiel.de or email@mwkrause.de (permanently private), telephone +49 431 880-7547
G. Sommer: gs@ks.informatik.uni-kiel.de, telephone +49 431 880-7540

[†]A set of points is said to be intrinsically n dimensional[20] in signal processing, or inD, if it lies entirely within a $(N - n)$-dimensional subspace of the embedding space of dimension N. This corresponds to the codimension in mathematics. In 3D, lines are i2D, planes are i1D, general surfaces and curves i0D.

Hestenes[16]. We will give a very brief introduction to it in section 2.
In section 3 we present the 3D monogenic signal and the isotropic quadrature filter derived from it in the Poisson scale space. Finally, in section 4 we apply the derived filter for motion estimation problems. We assume that the reader is familiar with basic motion estimation techniques. If not, we recommend to read one of these well done standard introductions[1, 2, 4, 17].

2. SOME NOTES ON GEOMETRIC ALGEBRA

Geometric algebra has a lot of useful applications and anyone interested may read the introductions referred to in section 1 as well as the papers on the topic on this website[‡]. Here, we will briefly state the things which are necessary to know to understand this paper.
We will deal with the geometric algebra $\mathcal{G}_4$ derived from $\mathbb{R}^4$ which consists of the following orthonormal basis

$$\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_4, \mathbf{e}_{12}, \mathbf{e}_{13}, \mathbf{e}_{14}, \mathbf{e}_{23}, \mathbf{e}_{24}, \mathbf{e}_{34}, \mathbf{e}_{123}, \mathbf{e}_{124}, \mathbf{e}_{234}, \mathbf{e}_{134}, \mathbf{e}_{1234} = I_4\}. \tag{1}$$

The 1 indicates the basis of a *scalar*, the $\mathbf{e}_i$ are the unit *vectors* of $\mathbb{R}^4$, the $\mathbf{e}_{ij}$ are unit *bivectors*, the $\mathbf{e}_{ijk}$ are unit *trivectors*, and $\mathbf{e}_{1234}$ or I_4 is called *pseudoscalar*. Any linear combination of different types of basis elements is called a *multivector*. The basis multivectors are spanning subspaces of $\mathcal{G}_4$ of different grade. The grade operator $\langle\cdot\rangle_i, 0 \leq i \leq 4$, gives the component of the ith grade of any multivector.
The most important thing about this algebra is its product, the *geometric product*. It is, for any two vectors a and b, defined to be

$$ab = a \cdot b + a \wedge b \tag{2}$$

The $\cdot$ product is called *inner product*. It is comparable to the standard scalar product. The $\wedge$ product is called *outer product* and is, in 3D, comparable to the standard cross product. There is a lot to know about the geometric product, however, the only we care about for now are the following simple rules for $i \neq j \neq k$:

$$\begin{aligned}
\mathbf{e}_i\mathbf{e}_i &= \mathbf{e}_i \cdot \mathbf{e}_i + \mathbf{e}_i \wedge \mathbf{e}_i = \mathbf{e}_i \cdot \mathbf{e}_i = 1 && (3)\\
\mathbf{e}_i\mathbf{e}_j &= \mathbf{e}_i \cdot \mathbf{e}_j + \mathbf{e}_i \wedge \mathbf{e}_j = \mathbf{e}_i \wedge \mathbf{e}_j = \mathbf{e}_{ij} = -\mathbf{e}_{ji} && (4)\\
(\mathbf{e}_i\mathbf{e}_j)\mathbf{e}_k &= \mathbf{e}_i(\mathbf{e}_j\mathbf{e}_k) = \mathbf{e}_i\mathbf{e}_{jk} = \mathbf{e}_i \wedge \mathbf{e}_{jk} = \mathbf{e}_{ijk} = -\mathbf{e}_{ikj} = \mathbf{e}_{kij} = -\mathbf{e}_{kji} && (5)\\
\mathbf{e}_i\mathbf{e}_{ij} &= \mathbf{e}_i(\mathbf{e}_i\mathbf{e}_j) = (\mathbf{e}_i\mathbf{e}_i)\mathbf{e}_j = \mathbf{e}_j && (6)\\
\mathbf{e}_i(\alpha\mathbf{e}_i + \beta\mathbf{e}_j) &= \alpha + \beta\mathbf{e}_{ij} = s && (7)
\end{aligned}$$

Equation (3) says that basis vectors square to 1 in $\mathcal{G}_4$. The outer product of a vector with itself is zero. Equation (4) on the other hand states that the inner product of orthogonal vectors is zero while the outer product gives a bivector. Note that the outer product is anticommutative, which means that $\mathbf{e}_i \wedge \mathbf{e}_j = -\mathbf{e}_j \wedge \mathbf{e}_i$. We see that the geometric product of two basis vectors gives a basis bivector. The product of three basis vectors gives a basis trivector, as shown in equation (5). We also see here that this product is associative. Again, swapping of two adjacent indices results in a sign change.

There are several reasons why we need bivectors, trivectors and pseudoscalars. In general these new algebraic entities in comparison to the vectors of $\mathbb{R}^4$ result in the necessary flexibility to model the required phase concept in higher dimensional space. At least for bivectors there is a very simple geometric interpretation which we understand when we regard equation (6). The bivector $\mathbf{e}_{ij}$ represents the operation needed to rotate $\mathbf{e}_i$ to $\mathbf{e}_j$, when applied from the right. So, a basis bivector represents a rotation by 90 degree in the plane spanned by the two vectors it was created from with the outer product. That is very similar to the complex unit i. This behavior is generalised in equation (7). It shows that the geometric product is distributive, but that is not the main point. We regard the product of a basis vector and a linear combination of basis vectors. Like in equation (6) this results in the operation needed to rotate $\mathbf{e}_i$ to $\alpha\mathbf{e}_i + \beta\mathbf{e}_j$, a linear combination of a scalar and a bivector, which we want to call a *spinor* s. A spinor is, thus, composed of an element of grade zero (scalar) and an element of grade two (bivector). When we associate again the $\mathbf{e}_{ij}$ with the imaginary unit i, the spinor looks

[‡]`www.ks.informatik.uni-kiel.de`

very much like an ordinary scaling-rotation as known from $\mathbb{C}$. In fact, for any fixed bivector B being a linear combination of basis bivectors, the spinor $s = \alpha + \beta \frac{B}{|B|}$[§] behaves like a scaling-rotation in $\mathbb{C}$. Or, in other words, every subalgebra generated by $\{1, \frac{B}{|B|}\}$ is isomorphic to $\mathbb{C}$. We will need this later.

3. A 3D ISOTROPIC QUADRATURE FILTER

3.1. Sketch of the Derivation

The basis to design this filter has been laid by Felsberg and Sommer[10] who described a 2D monogenic signal as an extension of the 1D analytic signal to images. This signal representation results from a generalisation of the Hilbert transform to higher dimensions.
Our work extends his approach to 3D image data. So, we can extract structure information described by the phase from image sequences in a video stream. It can of course be used also for e.g. extracting structure from 3D medical data.

Because of the great importance of the analytic signal in 1D signal processing, there have been several approaches to generalise especially the concept of the analytic phase to 2D[6, 15]. Felsberg[9] managed to accomplish this generalisation by embedding the signal in a vector space augmented by one dimension. Because this is the way we will follow here, our 3D signals have to be embedded into $\mathbb{R}^4$ and the local spectral representations have to be modelled in $\mathcal{G}_4$.
Because a full derivation would be out of the scope of this paper, we will just briefly sketch the idea. We embed the scalar signal $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^3$, into the geometric algebra $\mathcal{G}_4$ of $\mathbb{R}^4$ as a $\mathbf{e}_4$-valued function

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3) = f(x, y, z)\mathbf{e}_4. \tag{8}$$

This embedding enables us to generalise the methods of complex valued vector analysis and to define a set of Cauchy-Riemann equations for our problem. These equations play an important role for the derivation of the standard Hilbert transform and we succeeded in porting all steps necessary to higher dimensions. See the report[18] for all the details.

3.2. Definition of the Monogenic Signal

The chosen embedding results in the following convolution kernels,

$$r(\mathbf{x}) = \frac{\mathbf{x}\mathbf{e}_4}{\pi^2|\mathbf{x}|^4} \qquad \text{Riesz kernel,} \tag{9}$$

$$l_s = \frac{s}{\pi^2|\mathbf{x}+s\mathbf{e}_4|^4} \qquad \text{Poisson kernel,} \tag{10}$$

$$r_s(\mathbf{x}) = \frac{\mathbf{x}\mathbf{e}_4}{\pi^2|\mathbf{x}+s\mathbf{e}_4|^4} \qquad \text{conjugated Poisson kernel,} \tag{11}$$

where $\mathbf{x} = \sum_{i=1}^{3} x_i\mathbf{e}_i$ and $s \in \mathbb{R}$. Then the *monogenic signal*

$$\mathbf{f}_M(\mathbf{x}) = \mathbf{f}(\mathbf{x}) + (r(\mathbf{x}) * \mathbf{f}(\mathbf{x})) \tag{12}$$

is the generalisation of the analytic signal to 3D and

$$\mathbf{f}_{M_s}(\mathbf{x}) = (l_s(\mathbf{x}) * \mathbf{f}(\mathbf{x})) + (r_s(\mathbf{x}) * \mathbf{f}(\mathbf{x})) \tag{13}$$

is its smoothed version. Here $*$ denotes a convolution and s serves as scale parameter in a *monogenic scale space*[¶].
We see that the kernel of the Riesz transform (9) assumes the role that is inherent to the Hilbert transform kernel for the classical analytic signal. Note that both functions (12) and (13) are vector valued, while the convolution kernels (9–11) are either scalar or bivector valued. However, while important for the derivation and discussion, this can easily be implemented using standard matrix calculus on a computer.

[§]The magnitude of any multivector M is, as usual, the square root of the sum of squares of its components

[¶]The scale parameter is a result of the embedding of the original 3D data in an augmented vector space. While convolution of $\mathbf{f}(\mathbf{x})$ by the Poisson kernel results in a so-called *Poisson scale space*, which is similar to the Gaussian scale space, the monogenic scale space embeds also phase and orientation. Hence a unified scale space approach for local signal analysis emerges. On details concerning the new scale spaces please refer to this journal paper.[11]

3.3. Definition of Monogenic Energy and Phase

The deviation of the local spectral representations amplitude and phase from the complex valued analytic signal is well-known. In our approach we left the complex domain and adopted a Clifford valued domain. To model the local spectral representations in our case we imagine that the monogenic signal, equations (12) and (13), resulted from applying a spinor operation on a unit pulse in direction $\mathbf{e_4}$, just as any complex number on the unit circle results from a rotation of a real unit number. And we will take advantage from the isomorphism $\mathbb{C} \simeq \{1, \frac{B}{|B|}\}$, see section 2. The geometric product applied to two vectors results in a spinor and gives the relationship of these two vectors, as seen in equations (6, 7). Let us now calculate the logarithm of a spinor $s = \alpha + \beta\frac{B}{|B|}$. We do this by an adaptation of the complex logarithm[‖] and define

$$\log(s) = \log(|s|) + \frac{\langle s\rangle_2}{|\langle s\rangle_2|}\mathrm{atan}\left(\frac{|\langle s\rangle_2|}{\langle s\rangle_0}\right). \tag{14}$$

We see that the scalar part of equation (14) gives the (logarithmic) energy

$$\log(|s|) = \langle\log(s)\rangle_0 \tag{15}$$

of s, while the bivector part represents the spinor's phase

$$\arg(s) = \langle\log(s)\rangle_2. \tag{16}$$

Knowing this, we can assign an energy and phase to the monogenic signal (12)

$$\mathbf{f}_M(\mathbf{x}) = f(\mathbf{x})\mathbf{e}_4 + r_1(\mathbf{x})\mathbf{e}_1 + r_2(\mathbf{x})\mathbf{e}_2 + r_3(\mathbf{x})\mathbf{e}_3. \tag{17}$$

We see that it is vector valued. The $f(\mathbf{x})$ coupled with $\mathbf{e}_4$ is the original signal while the $r_i(\mathbf{x})$ are the Riesz components coupled with the other basis vectors. The monogenic signal can be considered as the action of a spinor s on a unit impulse oriented in the direction of $\mathbf{e}_4$. Hence, according to equation (7) we can regard the spinor $s = \mathbf{e}_4\mathbf{f}_M(\mathbf{x})$ as the rotation from $\mathbf{e}_4$ to $\mathbf{f}_M(\mathbf{x})$. So we can use the logarithm from equation (14) to calculate the energy and phase of the monogenic signal (17):

$$\log(\mathbf{e}_4\mathbf{f}_M(\mathbf{x})) = \log(|\mathbf{e}_4\mathbf{f}_M(\mathbf{x})|) + \frac{\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2}{|\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2|}\mathrm{atan}\left(\frac{|\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2|}{\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_0}\right). \tag{18}$$

Now, we *define* the local energy $E(\mathbf{x})$ and the local phase $\varphi(\mathbf{x})$ of the vector valued monogenic signal $\mathbf{f}_M(\mathbf{x})$ to be equal to the related quantities of the spinor presented.

$$\begin{aligned} E(\mathbf{x}) &= |\mathbf{f}_M(\mathbf{x})| \\ &= \exp(\log(|\mathbf{e}_4\mathbf{f}_M(\mathbf{x})|) \\ &= \exp(\langle\log(\mathbf{e}_4\mathbf{f}_M(\mathbf{x}))\rangle_0) \end{aligned} \tag{19}$$

$$\begin{aligned} \varphi(\mathbf{x}) &= \arg(\mathbf{f}_M(\mathbf{x})) \\ &= \langle\log(\mathbf{e}_4\mathbf{f}_M(\mathbf{x}))\rangle_2 \\ &= \frac{\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2}{|\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2|}\mathrm{atan}\left(\frac{|\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_2|}{\langle\mathbf{e}_4\mathbf{f}_M(\mathbf{x})\rangle_0}\right). \end{aligned} \tag{20}$$

The way we have defined the energy and phase is in fact a transformation of the monogenic signal to spherical coordinates, since it can be reconstructed by:

$$\mathbf{f}_M(\mathbf{x}) = E(\mathbf{x})e^{\varphi(\mathbf{x})} = |\mathbf{f}_M(\mathbf{x})|\exp(\arg(\mathbf{f}_M(\mathbf{x}))) \tag{21}$$

[‖]Let $w = u + iv$ and $z = re^{i\varphi}$. $w = \ln(z)$ means that $u = \ln(r)$ and $v = \varphi + 2k\pi, k \in \mathbb{Z}$.

So, $\mathbf{f}_M(\mathbf{x})$ consists of two orthogonal quantities, the scalar local energy describing the presence of local structure and the bivector valued local phase. The magnitude of the bivector is the arc tangent of the original signal and its "complex" component, just as for the classical Hilbert phase. This represents a rotation angle in the plane defined by the normalised bivector component of the phase. We would like to characterise the orientation of that plane in an easy way, like we proceed with normal vectors in $\mathbb{R}^3$. But in $\mathbb{R}^4$ the normal to a plane is a plane itself and not a vector. We see in equation (20) that every possible bivector is made up of $\mathbf{e}_4$ and $\mathbf{f}_M(\mathbf{x})$, thus being a pencil of bivectors with common $\mathbf{e}_4$. So, if we take out $\mathbf{e}_4$ by means of the inner product[**]

$$\begin{aligned}
&\frac{1}{|\langle \mathbf{e}_4 \mathbf{f}_M(\mathbf{x})\rangle_2|}(r_1(\mathbf{x})\mathbf{e}_{41} + r_2(\mathbf{x})\mathbf{e}_{42} + r_3(\mathbf{x})\mathbf{e}_{43}) \cdot \mathbf{e}_4 \qquad (22)\\
&= \frac{1}{|\langle \mathbf{e}_4 \mathbf{f}_M(\mathbf{x})\rangle_2|}(r_1(\mathbf{x})\mathbf{e}_1 + r_2(\mathbf{x})\mathbf{e}_2 + r_3(\mathbf{x})\mathbf{e}_3)\\
&= \frac{1}{\sqrt{r_1^2(\mathbf{x}) + r_2^2(\mathbf{x}) + r_3^2(\mathbf{x})}}(r_1(\mathbf{x})\mathbf{e}_1 + r_2(\mathbf{x})\mathbf{e}_2 + r_3(\mathbf{x})\mathbf{e}_3)
\end{aligned}$$

we have got a vector representation of the bivector in the subspace spanned by $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$, which is the domain of the original signal. This is, not very surprising, the normalised Riesz component of the monogenic signal. We can say that the normalised vector of the Riesz component gives the orientation of the structure, see the report[18] for details.

Let us sum things up. The magnitude of the monogenic signal $|\mathbf{f}_M(\mathbf{x})|$ is an indicator for the presence of local intrinsically 1D structure, which are planes in 3D. The phase is a vector field. The arc tangent of the magnitude of the Riesz component divided by the signal itself gives the classical analytic phase, thus describing the nature of this structure. The direction of the vector of the Riesz component additionally gives the orientation of the structure, being normal to the planes described. The monogenic signal needs no steering of oriented filters, it delivers the right orientation of the structures for free. This results from the fact that the Riesz kernel, equation 9 is nothing else than a spherical harmonic.

3.4. The Spherical Quadrature Filter

When we apply the Poisson kernel (10) and the conjugated Poisson kernel (11) to a function $\mathbf{f}(\mathbf{x})$ we obtain a low-pass filtered version of the monogenic signal, $\mathbf{f}_{M_s}(\mathbf{x})$, as shown in equation (13). If we apply kernels with two different scales, a coarse scale c and a fine scale f, with $c, f \in \mathbb{R}_0^+, c > f$, we get a bandpass filtered monogenic signal,

$$\begin{aligned}
\mathbf{f}_M^{c,f}(\mathbf{x}) &= \mathbf{f}_M^c(\mathbf{x}) - \mathbf{f}_M^f(\mathbf{x})\\
&= [(l_c(\mathbf{x}) - l_f(\mathbf{x})) * \mathbf{f}(\mathbf{x})] + [(r_c(\mathbf{x}) - r_f(\mathbf{x})) * \mathbf{f}(\mathbf{x})]\\
&= [l_{c,f}(\mathbf{x}) * \mathbf{f}(\mathbf{x})] + [r_{c,f}(\mathbf{x}) * \mathbf{f}(\mathbf{x})]. \qquad (23)
\end{aligned}$$

We call $l_{c,f}$ *Difference-of-Poisson* kernel (DOP) and $r_{c,f}$ *Difference-of-Conjugated-Poisson* kernel (DOCP). In practice, it is very important to use these kernels to calculate the monogenic signal, since the Hilbert transform, as well as its generalisation to higher dimensions, only gives good results for narrow-banded signals.

4. OPTICAL FLOW: AN APPLICATION OF THE MONOGENIC SIGNAL

In this section we want to study how to use the monogenic signal of 3D data for motion estimation. It is assumed that the reader is familiar with the basic concepts of flow field estimation. As an introduction to this topic we can recommend[1, 2, 4, 17].

We will skip the use of geometric algebra here, since this section is concerned with implementation issues.

[**]We use an identity not mentioned in section 2. For three vectors a, b, c the following is true: $a \cdot (b \wedge c) = (a \cdot b)c - (a \cdot c)b$

4.1. Basic Estimation Technique

We have claimed that we wanted to use the phase vector as a *motion constraint*. The idea of the motion constraint is initially from Horn and Schunck[17] who used first order derivatives to constrain the motion field. Let us assume that the intensity I of a moving structure is preserved during time,

$$I(\mathbf{x}, t) = I(\mathbf{x} - \mathbf{v}t, 0), \tag{24}$$

where $\mathbf{x} = (x, y)^T$ is a point in the image and $\mathbf{v} = (u, v)^T$ is the local velocity. A first order Taylor expansion of (24) leads to

$$\nabla I(\mathbf{x}, t) \cdot \mathbf{v} + I_t(\mathbf{x}, t) = 0, \tag{25}$$

which is called the *gradient constraint equation.* There are two unknowns in equation (25), so further constraints are needed to solve for $\mathbf{v}$. Horn and Schunck combined eq. (25) with a global smoothness term, thus minimising the functional

$$G = \int (\nabla I \cdot \mathbf{v} + I_t^2) + \alpha(|\nabla u|_2^2 + |\nabla v|_2^2) d\mathbf{x}. \tag{26}$$

This can be solved by using iterative equations, so the image velocity is gained by

$$\begin{aligned} u^{k+1} &= \bar{u}^k - \frac{I_x[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \\ v^{k+1} &= \bar{v}^k - \frac{I_y[I_x\bar{u}^k + I_y\bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2}. \end{aligned} \tag{27}$$

In our further evaluation the smoothness regularisation is set to $\alpha = 0.5$ as suggested in[2]. As for the iteration, we used $\mathbf{v}^0 = (0, 0)^T$ and the maximal number of iterations is set to $k_{max} = 100$. Equation (27) may even terminate sooner when both u_k and v_k exceed a threshold ϵ, more below. The quantities $\bar{u}$ and $\bar{v}$ are averages over a certain neighborhood, normally 3×3. Instead of using arithmetic averages, we apply a (3×3)-median filter, as suggested in[3].

The difference of our phase approach to the gradient based approach described here is that we do not use the gradient $(I_x, I_y, I_t)^T$ as a motion constraint. Instead, we use the orientation of the monogenic phase, $(r_1, r_2, r_3)^T$ or its normalised version, see equation (22). Since the orientation only makes sense where structure – lines or edges – is present in the image, our approach calculates a sparse motion field.

4.2. The Advantages of Phase Constraints

There are a few works that deal with phase based motion estimation[4, 12]. But they all suffer from not being able to get the phase from three dimensional image data in a consistent way. We have presented a means to solve this.
By using phase information it is much easier to extract relevant structure from the data than by using the gradient. Figure 1 illustrates this fact. The figure shows (x, t)-slices from (x, y, t) gray-value image data. In the left column we see a gray value structure that moves in x-direction with growing t. The velocity of this structure is 1 pixel per frame. The difference between the top four and bottom four plots is that for the first ones the intensity of the moving structure remains constant while for the latter ones it increases with growing t.
The vector fields printed over the left-column gray-value images are the gradient field or the phase vector field, giving local orientation. We have normalised both vector fields to just show the orientation information. The magnitude of the phase is given in the right column, as is the magnitude of the gradient.

Let us first regard the upper four plots of figure 1 where the structure remains constant in intensity. We see that both the gradient and phase work fine to determine the location and orientation of edges. Of course there are problems around the borders of the gray value images for both approaches. There are many more arrows in the phase plot. That is, because we use a convolution mask of $(13 \times 13 \times 13)$ pixels. But we can reduce the number of arrows when we just regard the locations where the magnitude of the phase indicates relevant structure. That is $-\pi/2$ and $\pi/2$ for edges and 0 or $(\pi, -\pi)$ for lines. Note that the phase vector flips as the magnitude of the phase traverses 0 for the line structure. But that does not affect the orientation of the structure, since orientation is only defined to lie within $[0..\pi]$.

Let us now look at the bottom four plots. We see that both gradient and phase give nearly the right orientation of the edges. Of course there is a slight deviation due to the intensity changes. But we also see that there are some more gradient arrows within the moving structure pointing downwards. These values are completely unusable as a motion constraint, so we have to eliminate them. This can be done using a threshold on the magnitude of the gradient. But in real images, using thresholds is always a trade-off.
For the phase, on the other hand, it is easy to pick the locations where the edges are, because still the phase magnitude is either $-\pi/2$ or $\pi/2$. Note that the two plots showing the magnitude of the phase are quite similar, while the plots showing the magnitude of the gradient differ a lot.
It is eye-catching that the behavior of the phase vector within the moving structure differs from the one in the above plot although the magnitude of the phase is similar. Still the energy of the even part of the monogenic signal – that is the intensity function convolved with the DOP – is much bigger than the energy of the odd part – the intensity function convolved with the DOCP – resulting in a phase magnitude close by 0. On the other hand, since the structure is not truly intrinsically 2D[††] anymore, the odd part of the monogenic signal gives a wrong orientation. We conclude from this that we only want to trust edge structures for motion estimation.

You may point out that it is an unfair method to compare the phase with the gradient since the phase is calculated from a smoothed version of the image: the wrong orientation of the gradient within the structure is certainly due to the aperture effect and the locality of the gradient. On the other hand, a smoothing of the image used for the gradient would have resulted in a gradient vector field quite similar to the phase vector field in the bottom left, but without the phase information. It would have been even harder to extract the relevant gradient vectors for motion estimation.

4.3. Performance on Simple Data

We want to compare the performance of the monogenic phase and the gradient as motion constraint on simple artificial data. The simplest motion a structure can assume is one with constant velocity along a fixed axis. So we want to study a vertical edge which moves with constant velocity in x-direction. The 3D image data is generated using bilinear interpolation.
For the gradient based approach we simply use MATLAB's `gradient` function. For the phase-based approach we use the DOP and DOCP introduced in section 3.4 with $c = 2$ and $f = 1$. The kernel has a support of $(13 \times 13 \times 13)$ pixels.
Figure 2 shows a basic experiment to evaluate the accuracy of the two approaches. The velocities regarded range from 0 to 5.67 pixels per frame.

Figure 2, top row, shows the measured velocities using the different constraints directly compared with the true velocity. We see that, regarding this graphs, the phase constraint produces good results for the whole range of velocities while the gradient constraint fares poorly for velocities above 1 pixel/frame. The latter is not too surprising since we do not use downsampling for the gradient. While we still can compare the phase and gradient for velocities below 1 pixel/frame, we also learn that we would have to downsample thrice for the gradient to be able to handle the velocities the phase-based approach can estimate.
The error of the phase approach decreases with decreasing ϵ while the result produced by the gradient seem to change not at all in the last three diagrams. Let us have this a little more precise: switch to the middle row of this figure.

The middle row displays the relative error of the different constraints for different values of ϵ. Note the logarithmic scale of the axis displaying the error. We see that for both approaches the error has a minimum in the vicinity of 1 pixel/frame. That is due to the generation of our test data: there is now interpolation necessary for this velocity. The phase-based approach almost always produces better results than the gradient based approach.

The bottom row of figure 2 compares the number of iterations needed for the two approaches for the three different thresholds ϵ. We see that with the gradient method the algorithm quite soon uses the maximum number

[††]with respect to the (x, y, t) gray-value image data. Of course a line would be i1D in 2D images.

of iterations, which explains why the gradient related graphs change so little in the top and middle row of figure 2. The phase-based approach shows a much better convergence rate.

Even though the output of the phase approach is superior to that of the gradient method for this setup we must not forget that the first needs the 3D data to be convolved with a $(13 \times 13 \times 13)$ pixels filter mask, while the latter is just a differential quotient and can be implemented very efficiently. So the gradient approach is very local and fast while the phase approach can be scaled arbitrarily to produce stable results but is much slower. Remember that we stated in section 4.2 that we did not presmooth the images for the gradient since this introduces more movement to the image, which is not easy to handle, since the gradient does not say anything about the quality of an image location for motion estimation. For the phase vector, in contrast, we can use the magnitude of the phase to decide which orientation estimates stick to relevant structures that we want to trust for motion estimation.

Figure 3 shows a portion of figure 2 with various levels of Gaussian noise. The noise levels are attached to the plots in the top graph and are the same for the plots in the bottom graph. When we say $n\%$ noise, we mean that the standard deviation of the Gaussian distribution added to the voxels of the test data is just $n\%$ of the maximum gray value.
We see that as well the relative error and the number of iterations needed to converge grows with growing noise level, which is of course exactly what we expect. The convergence rate is for all noise levels better than for the gradient with no noise. The relative error is below 5% for noise levels up to 4%. For a noise level of 32% the algorithm still converges with less than 80 iterations and about 10% error.

5. SUMMARY

Let us briefly summarise the strengths and weaknesses of our approach: We can calculate a phase-based motion constraint field in a consistent way without using phase derivatives. The quadrature filter does not need steering since we use an isotropic approach, four 3D convolutions are enough to get full orientation, energy, and phase information. Additionally, the derivation of the filter yields a scale parameter that belongs to the Poisson scale space just recently discovered.
A filter of this design has many useful applications. We have shown that, for example, the acquired phase vector field can serve as a motion constraint. It has been tested with the Horn and Schunck algorithm, but can in principle be used by any other algorithm of this kind. It is superior to the standard gradient motion constraint in a way that we can deduce from the local phase information where we have relevant structure. This helps to determine where we want to trust the output of the Horn and Schunck algorithm, which normally produces a smoothed, dense flow field. Also, on the artificial data we used for testing, the phase motion constraint resulted in a relative error for the estimated velocity nearly a degree of magnitude below that one archieved using the gradient constraint.
On the other hand, since we perform several 3D convolutions, we need a whole stack[‡‡] of images to calculate the flow field for the one in the middle. While this increases stability by smoothing over time it also results in a lag when we want to do online flow field estimation. Another point is that the convolutions are non-separable and need a lot of computational power. For offline flow field estimation, speed can be improved to a great extend by using the Fourier representation of the kernels presented, see this report[18], along with the convolution theorem.

[‡‡]Our current experiments are performed with a filter mask of $13 \times 13 \times 13$ pixels.

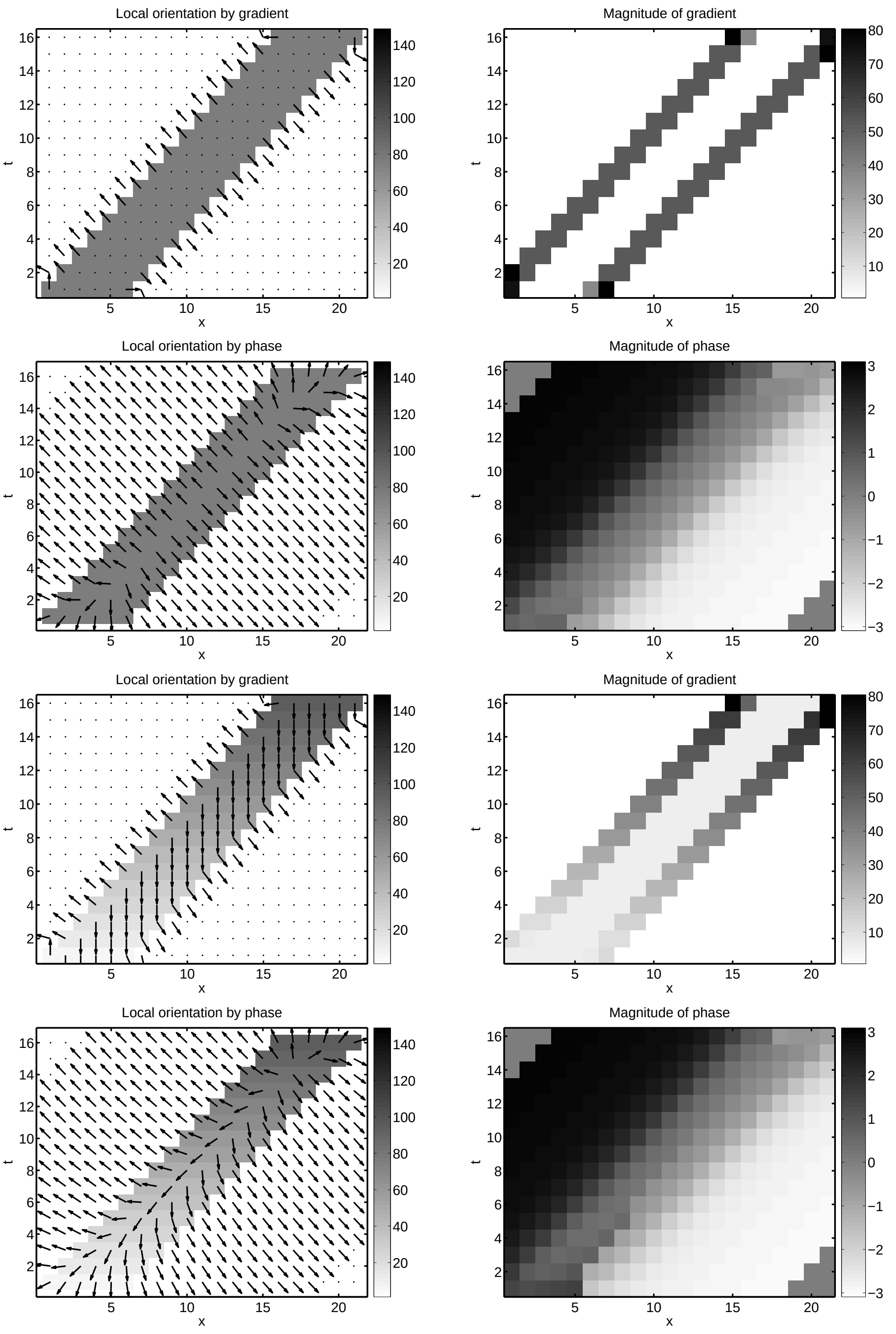

Figure 1. *Comparison of gradient and phase for usability as a motion constraint for a moving structure with fixed velocity and fixed intensity (upper four plots) or growing intensity (lower four plots). It is striking, that the magnitude of the phase is nearly the same for both situations while the magnitude of the gradient differs greatly. So the phase is a reliable indicator for structure. See text for details.*

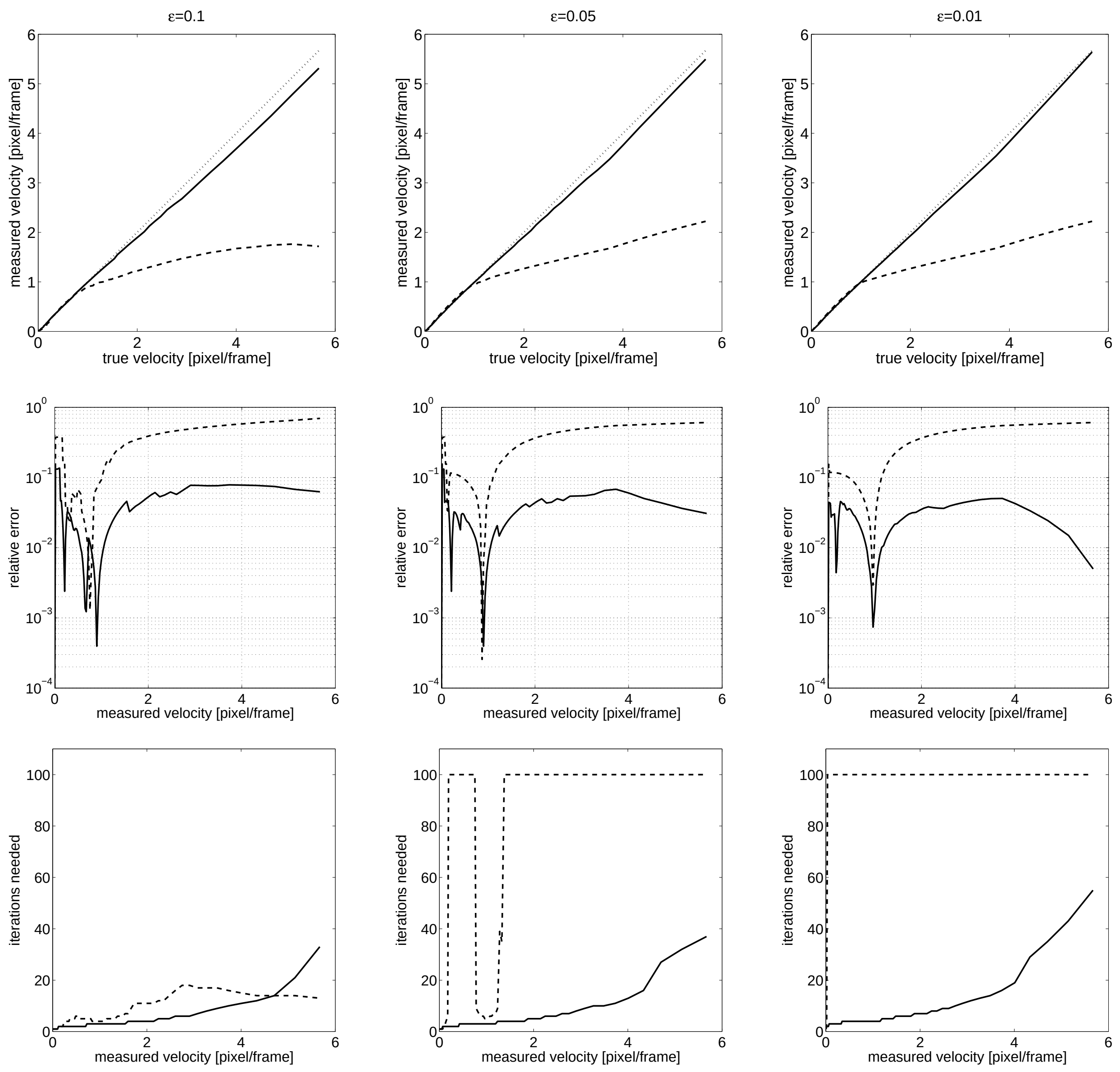

Figure 2. *Comparison of the phase (solid line) and the gradient (dashed line) as motion constraint. We use the Horn and Schunck algorithm to estimate the velocity of a vertical edge moving with constant velocity.*
The top row shows the estimated velocity against the true velocity (dotted line) for three different convergence thresholds ϵ. The values for epsilon given in the top row apply for the whole column. The middle row shows the relative error of the estimated velocity. Note that the scale of the ordinate is a logarithmic one. In the bottom row we see the number of iterations needed for the Horn and Schunck algorithm to converge below epsilon for each of the two motion constraints. Tha maximum number of iterations is $k_{max} = 100$.

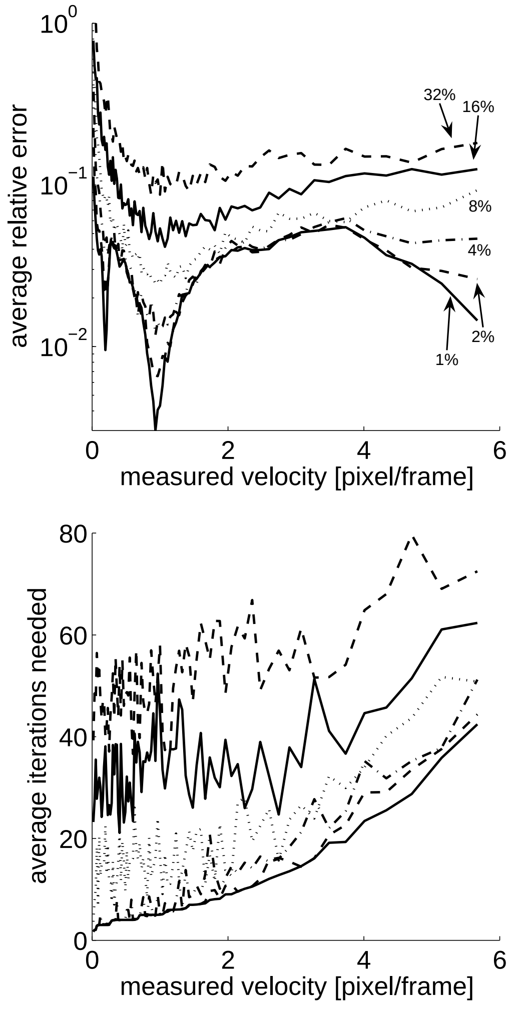

Figure 3. *Bottom two graphs of the right column of figure 2 with noise and without gradient: average relative error (top) and average number of iterations needed (bottom) for the phase motion constraint for noisy data and a threshold* $\epsilon = 0.01$*. The average was taken over* 50 *repetitions of the experiment. The noise level is attached to the curves in the top graph and is the same for the curves in the bottom graph. Keep in mind that the scale of the ordinate in the top graph is logarithmic.*

REFERENCES

1. J.L. Barron, S.S. Beauchemin, and D.J. Fleet. On optical flow. *6th Int. Conf. on Artificial Intelligence and Information-Control Systems of Robots, Bratislava, Slovakia*, pages 3–14, Sept. 12–16 1994.
2. J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt. Performance of optical flow techniques. *CVPR*, 92:236–242, 1992.
3. F. Bartolini and A. Piva. Enhancement of the Horn and Schunck optic flow algorithm by means of median filters. *Proceedings 13th International Conference on Digital Signal Processing DSP97, Santorini, Greece*, pages 503–506, July 2–4 1997.
4. S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–467, 1995.
5. D. Boukerroui, J. A. Noble, and M. Brady. On the choice of band-pass quadrature filters. *Journal of Mathematical Imaging and Vision*, 21 (1):53–80, July 2004.
6. T. Bülow. Hypercomplex spectral signal representations for the processing and analysis of images. Technical Report Number 9903, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, August 1999.
7. C. Doran and A. Lasenby. Physical applications of geometric algebra. www.mrao.cam.ac.uk/∼clifford/ptIIIcourse, 2002.
8. L. Dorst, S. Mann, and T. Bouma. GABLE: A MATLAB tutorial for geometric algebra. www.cgl.uwaterloo.ca/∼smann/gable/, August 2000.
9. M. Felsberg. Low-level image processing with the structure multivector. Technical Report Number 0203, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, März 2002.
10. M. Felsberg and G. Sommer. The monogenic signal. *IEEE Transactions on Signal Processing*, 49(12):3136–3144, December 2001.
11. M. Felsberg and G. Sommer. The monogenic scale-space: A unifying approach to phase-based image processing in scale-space. *Journal of Mathematical Imaging and Vision*, 21:5–26, 2004.
12. D.J. Fleet and A.D. Jepson. Computation of component image velocity from local phase information. *IJCV*, 5(1):77–104, 1990.
13. G. Granlund and H. Knutsson. *Signal Processing for Computer Vision.* Kluwer Academic, Dordrecht, 1995.
14. S. Gull, A. Lasenby, and C. Doran. Imaginary numbers are not real – the geometric algebra of spacetime. Technical report, Cavendish Laboratory, Cambridge, February 1993.
15. S.L. Hahn. *Hilbert Transforms in Signal Processing.* Artech House, Inc., Norwood, MA 02062, 1996.
16. D. Hestenes and G. Sobczyk. *Clifford algebra to geometric calculus.* Kluwer Academic Publishers, Dordrecht, 1992. 1st. ed 1984, Reprinted with corrections.
17. B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1980.
18. M. Krause and G. Sommer. A generalisation of the analytic signal for 3D data and its applications. Technical report (to be published), Institute of Computer Science and Applied Mathematics, University of Kiel, Germany, 2005.
19. C. Perwass and D. Hildenbrand. Aspects of geometric algebra in euclidean, projective and conformal space. Technical Report Number 0310, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, September 2003.
20. C. Radon and E. Barth. Fundamental limits of linear filters in the visual processing of two dimensional signals. *Vision Research*, 30:1111–1117, 1990.