J Math Imaging Vis (2017) 58:349–381
DOI 10.1007/s10851-017-0715-7

Spherical Tensor Algebra: A Toolkit for 3D Image Processing

Henrik Skibbe[1] · **Marco Reisert**[2]

Received: 26 July 2016 / Accepted: 21 February 2017 / Published online: 7 March 2017

Abstract With the advent of novel 3D image acquisition techniques, their efficient and reliable analysis becomes more and more important. In particular in 3D, the amount of data is enormous and requires for an automated processing. The tasks are manifold, starting from simple image enhancement, image reconstruction, image description and object/feature detection to high-level contextual feature extraction. One important property that most of these tasks have in common is their covariance to rotations. Spherical Tensor Algebra (STA) offers a general framework to fulfill these demands. STA transfers theories from mathematical physics and harmonic analysis into the domain of image analysis and pattern recognition. The main objects of interest are orientation fields. The interpretations of the fields are manifold. Depending on the application, they can represent local image descriptors, features, orientation scores or filter responses. STA deals with the processing of such fields in the domain of the irreducible representations of the rotation group. Two operations are fundamental: the extraction/projection of the features by convolution-like procedures and the nonlinear covariant combination by spherical products. In this paper, we propose an open-source toolbox that implements, in addition to fundamental STA operators, advanced functions for feature detection and image enhancement and makes them accessible to the 3D image processing community. The core features are implemented in C (CPU and GPU) with APIs in C++ and MATLAB. As examples, we show applications for medical and biological images.

Keywords Biomedical 3D image processing · Rotational invariance · Spherical tensors · 3D feature detection · Bi-spectrum

1 Introduction

The analysis of three-dimensional images has gained more and more importance in recent years. In particular, new acquisition techniques in the medical and biological sciences produce an enormous amount of 3D data calling for automated analysis. In this article, we show how the harmonic analysis of the 3D rotation group offers a convenient framework for rotation covariant image processing and analysis.

A typical rotation covariant processing pipeline may considered as a kind of image filter. An example is illustrated in Fig. 1. First, local image features are extracted. These features are then processed in order to gather, combine and generate relevant information. Finally, one or more resulting images are created. We call such a pipeline *rotation covariant*, if the same filter, when applied to a rotated version of the input image, leads to a rotated version of the output. Since Cartesian tensors are rotation covariant by definition, they are widely used as the basic tool to design covariant filters.

Up to now, most algorithms rely on "low"-order features from Cartesian tensors like local intensities, intensity gradients or second-order derivatives and their products. For example, consider a lesion detection/segmentation problem in a T_1-weighted magnetic resonance image. A typical approach would be to gather certain kind of features for each

✉ Henrik Skibbe
skibbe-h@sys.i.kyoto-u.ac.jp

Marco Reisert
marco.reisert@uniklinik-freiburg.de

[1] Ishii-Lab, Department of Systems Science, Graduate School of Informatics, Kyoto University, Kyoto, Japan

[2] Medical Physics, Faculty of Medicine, University Freiburg, Freiburg im Breisgau, Germany

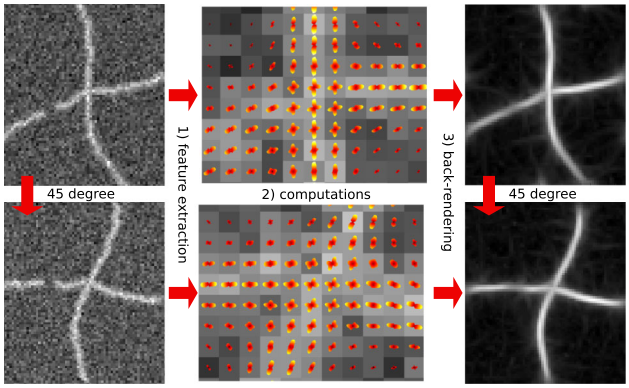

Fig. 1 Typical pipeline consists of three steps: (1) feature extraction: local image features are represented in an angular-dependent manner in terms of spherical tensors. (2) computations are performed in the spherical tensor domain (here a anisotropic smoothing, we see the crossing in the center). (3) The results are transformed back into an interpretable image

voxel, an example is a Laplacian- or a Gaussian pyramid, to determine the distribution of such features in a healthy state. This distribution can be used to find probabilities for certain voxels to contain a lesion or not. Instead of using zero-order features, like the Laplacian-pyramid, higher-order features, as the smoothed intensity gradient magnitudes (1-order features) or the eigenstructure of the Hessian matrix and the structure tensor (2-order features), can improve the performance. However, features of order 3 and beyond are rarely used because it is not trivial to identify linearly independent higher-order features [105,118]. The redundancies in the Cartesian tensor representation are cumbersome and hard to be handled operationally [105].

This article proposes a unified framework based on spherical tensors, which incorporates higher-order features in a systematic way. Spherical tensors are a common, exhaustedly studied object in the angular momentum theory. However, they are, so far, rarely used in the 3D image processing community. A reason for this might be the fact that for spherical tensors Cartesian directional information is, in contrast to Cartesian tensors, obscured by a complex-valued unitary coordinate transform between the Cartesian and the spherical representation. Moreover, unlike for tensors in Cartesian space, which can simply be extended to arbitrary dimensions, spherical tensors are strongly connected to the representation of the 3D rotation group and thus only exist in 3D space.

The main difference between spherical and 3D Cartesian tensors is that spherical tensors have a fixed number of indices no matter which order they are. In Cartesian representation, the number of indices is determined by the order of the tensor. A Cartesian tensor of order n has n indices of fixed dimension of 3. In spherical representation, there exists only one index. With growing order, the dimension of this index is growing as well: A spherical tensor of order n has one index of dimension $2n+1$. This property is, from an algorithmic point of view, a strong advantage. We can always deal with high-order tensors in the same way as with low-order tensors. This eases an optimized implementation.

Cartesian tensors, with their many indices, are reducible in the sense that indices can be folded together to form other indices, which still have a valid rotation behavior. An example is the trace. It let vanish two indices of a Cartesian tensor. The result is a new tensor with a reduced number of indices. For instance, the rotation corresponding to the trace of a second-order Cartesian tensor is trivial: It is the identity transformation.

For spherical tensors, since they only have one index, such operations, like the trace, do not exist. There exists no linear transformation (apart from the orthogonal/unitary ones) that can qualitatively change the rotation behavior of a spherical tensor. Spherical tensors are called irreducible. A consequence of the irreducibility is that spherical tensors are a kind of Fourier coefficients of spherical functions. They are, dependent on their order, associated with attributes like *rotation symmetry*, *sharpness*, *richer in details*, but also attributes like *noise* or *less relevant (high frequency) details*, the typical characteristics of image Fourier coefficients. We talk about the details later in this article.

In summary, it can be stated that in comparison with ordinary Cartesian tensor analysis, the algorithms and the handling are operationally much more clearer for spherical tensors. The combinatorial issues arising with Cartesian tensors are eliminated by the group representation theory of 3D rotations, though the involved representation theory is not easily accessible for the non-experienced. However, it allows the creation of efficient algorithms, particularly when higher-order tensors are involved.

In this work, we propose an open-source toolbox which covers all basic operations involved in spherical tensor algebra. The focus on the toolbox lies on the feature and object detection in large volumetric, biomedical images, and on the processing of tensor-valued images like diffusion-weighted MRI[1]. The toolbox is written in C and has a C++ and MATLAB/Octave API. A parallel CPU and GPU implementation is available as well. A repository with the source code is publicly available [98].

The article is divided into five sections. First, in Sect. 2, we introduce the basic theoretical concepts. We show the relationship between spherical and ordinary Cartesian tensors. We introduce the notion and properties of spherical tensors and address their relation to Fourier analysis.

In Sect. 3, we introduce orientation and spherical tensor fields. We propose two fundamental operations: spherical products and spherical derivatives, both important for image feature extraction and image filtering. Further, we introduce

[1] Magnetic Resonance Imaging

tensor-valued basis functions for the efficient computation of rotation covariant and invariant features. In focus are a Gauss–Laguerre basis and a Gabor basis. Both are known to be important in pattern analysis [13,16,24,27,39,40,47,56,57,63,66,67,69,74,77,89,92,104,118,120].

Section 3 comes with two novel contributions. (1) We transfer knowledge about symmetries of angular momentum states known in angular momentum theory to a feature description problem: We show how to avoid redundancies in spherical bi-spectra using a set of associativity rules in tensor products; this problem has, as far as we know, not been addressed so far. Since this saves both memory and computation time, it is, from an application point of view, an important aspect. (2) We also show how the diffusion equation in the position/orientation space can be efficiently solved via STA. Details about this topic have been presented in a technical report, which is available online [87].

Implementation concepts and implementation details are explained in Sect. 4. Finally, Sect. 5 reviews applications of STA, or extends, in the case of steerable deconvolution [86], existing algorithm from 2D to 3D. In this section, we give implementation examples, which can directly serve as a skeleton for biologically or medically relevant feature detection or image processing tasks.

1.1 Related Work

In two dimensions, the representation of orientation and tensor fields in terms of circular harmonics (or, the irreducible representations of $SO(2)$) is relatively simple and quite frequent in the literature [25,45,53–55,64,79,80,96,119]. Complex calculus offers a well-founded background: The ordinary Cartesian partial derivatives ∂_x, ∂_y are replaced by the complex derivatives $\partial_z = (\partial_x - \mathbf{i}\partial_y)/2$ and $\partial_{\overline{z}} = (\partial_x + \mathbf{i}\partial_y)/2$. In [82,103], three-dimensional derivative operators are introduced that behave similar to complex derivatives, that is, they are compliant with the rotation behavior of spherical harmonics in 3D. In Refs. [22,23], the Fourier transform of $SE(3)$ is used in the context of engineering applications. For the efficient computation of the $SE(3)$-convolution, functions are expressed in terms of the unitary irreducible representations (UiR) of $SE(3)$. In the context of line and contour enhancement in 2D, there are various works about orientation fields [30,31,113,114]. It can be used to set up a scale space theory. More recently, extensions to 3D of these concepts appeared [29]. While the applications in 2D are typically related to feature detection and image enhancement, the 3D extension offers a new application field: the processing of diffusion-weighted magnetic resonance images (DWI). In DWI, the acquired measurements are already functions on $\mathbb{R}^3 \times S_2$. Based on the directional dependency of water diffusivity in fibrous tissue of the human brain, it is possible to reveal underlying connectivity information. One of the main challenges in DWI is the estimation of so-called fiber/diffusion orientation distributions. There are numerous methods for estimating orientation distributions: classical Q-ball imaging [111], constrained spherical deconvolution [108], proper probability density estimation [2,11,18,109] and spatially regularized density estimations for tensor-valued images [9,17,49,84,90,110]. Most of the employed algorithms rely on tensorial or spherical harmonic representation of the orientation distributions. However, most of the algorithms for orientation distribution estimation that consider the local surrounding of a voxel, i.e., using inter-voxel information, rely on a discretization of the two-sphere [10,28,29,84].

The work on classical, rotation invariant 3D features, like 3D extensions of SIFT and SURF, is manageable and focused on solving point matching and registration problems. References [20,21] have proposed the N–D Sift descriptor, which has been used for the co-registration of volumetric medical 3D and 3D+time images. This includes the 3D-SIFT descriptor of [4]. The proposed 3D-SIFT descriptors have also been used for the registration of volumetric spectral OCT[2] images of the retina [73], or the co-registration of 3D + time CT scans of lung vessel trees [75,76]. Further applications on object recognition include the scanning of volumetric CT scans of bags in airports for security reasons [38] and a voting-based classification of objects in volumetric images [65] based on 3D SURF. In contrast, the usage of STA for the rotation invariant feature detection has remarkably increased the last years; for instance [34–37,59,60,70,81,91,93,94,99,102,103]. We will introduce examples later in the application section.

2 From Cartesian to Spherical Tensors

Cartesian tensors are often used to describe spatial properties of physical systems. In image analysis, Cartesian tensors are an indispensable tool for representing image characteristics with respect to the Cartesian coordinate system. Typical 3D Cartesian tensors are image gradients, local Hessian matrices or structure tensors [3,12,41,52]. 3D Cartesian tensors clearly exhibit directional information with respect to the Cartesian coordinates. This is particularly true for low-rank Cartesian tensors or tensors with certain symmetries. For instance, the eigensystem of a Hessian matrix directly represents the local image main curvatures in terms of directions and magnitudes, which is a widely used feature for the numerical computation of maxima of lower-order steerable image filters [3,5,42].

Every Cartesian tensor is associated with an order $n \in \mathbb{N}_0$. A Cartesian tensor $\mathbf{T}^n$ in 3D of order n is a mathematical

[2] Optical Coherence Tomography.

object with 3^n independent values $t^{(n)}_{j_1,\ldots,j_n} \in \mathbb{R}$ with $j_k \in \{0, 1, 2\}$.

We say that it has 3^n degrees of freedom (DOF). An order 0 tensor is a scalar. Tensors of order one and two are often written as vectors or matrices, respectively, whereas

$$\mathbf{T}^1 = \begin{pmatrix} t_0 \\ t_1 \\ t_2 \end{pmatrix} \qquad \text{and} \qquad \mathbf{T}^2 = \begin{pmatrix} t_{00} & t_{01} & t_{02} \\ t_{10} & t_{11} & t_{12} \\ t_{20} & t_{21} & t_{22} \end{pmatrix}. \quad (1)$$

Let $\mathbf{R}(g) \in \mathbb{R}^{3\times 3}$ be the standard representation of the rotation group. With g, we denote an element of the 3D rotation group $SO(3)$. Just think of it as a given triple of Euler angles (θ, ϕ, ψ). The characteristic of a Cartesian tensor is its behavior with respect to rotations. If the rotation acts in a certain way on the numbers $t^{(n)}_{j_1,\ldots,j_n}$, we say it is a tensor. For first- and second-order tensors, these actions can be expressed in ordinary matrix calculus by

$$(g\mathbf{T}^1) := \mathbf{R}(g)^T\mathbf{T}^1 \qquad \text{and}$$
$$(g\mathbf{T}^2) := \mathbf{R}(g)^T\mathbf{T}^2\mathbf{R}(g).$$

Here $g\mathbf{T}$ denotes the 'action' of the rotation group. For the general case, we need index representations:

$$(gt^{(n)})_{i_1,\ldots,i_n} = \sum_{j_1,\ldots,j_n} R(g)_{i_1,j_1} \ldots R(g)_{i_n,j_n} t^{(n)}_{j_1,\ldots,j_n}, \quad (2)$$

that is, all components $t^{(n)}_{j_1,\ldots,j_n}$ do "mix" under a rotation.

In Cartesian, as well as in spherical tensor calculus, there are two basic operations that combine tensors or create new tensors: derivatives and products.

Differentiation is a natural way to map 3D image information to 3D Cartesian tensors. For instance, given an image $I : \mathbb{R}^3 \to \mathbb{R}$ and successively differentiating along the Cartesian X-,Y-, and Z-axis creates an $(n+1)$-th order derivative which is given by

$$t^{(n+1)}_{i_1,\cdots,i_{n+1}} := \frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_{n+1}}} I. \quad (3)$$

The results of (3) transform according to (2) and thus are 3D Cartesian tensors of order $(n + 1)$.

Derivatives of different orders can be combined with tensor products to form features like the inner product of a gradient, an indicator for the presence of edges, or the trace of a Hessian matrix, a measure for blob-like structures.

In Cartesian tensor calculus, several ways exist to combine tensors. The outer product (the Kronecker-products) multiplies all elements of a tensor $\mathbf{T}^n$ with the elements of a tensor $\mathbf{T}^m$. This results in a new tensor of the order $n+m$. For example, the outer product $t^{(2)}_{ij} := t^{(1)}_i t^{(1)}_j$ creates a matrix out of two vectors.

There are two tensors which are fix points under rotations: the symmetric delta tensor $\delta_{ij} := \delta(i - j)$, which corresponds to the identity matrix, and the antisymmetric epsilon tensor ϵ_{ijk} (see Definition 6 in appendix). Due to their special rotation behavior, they can be used to build tensors out of existing ones. With the delta tensor, we can determine the sum over a pair of indexes (ij). This operation is called a tensor contraction, or the trace of a tensor. It reduces a tensor order by 2. An example is the trace of a second-order tensor $tr(\mathbf{T}^2) := \sum_{i,j} \delta_{i,j} t^{(2)}_{i,j}$. On the other hand, the combination of the product and the delta tensor increases the tensor rank by two. With $t^{(2)}_{ij} := \delta_{ij} t^0$, we obtain the second-order tensor $\mathbf{T}^0\mathbf{I}_{3\times 3}$ out of the zero-order tensor $\mathbf{T}^0$, where $\mathbf{I}_{3\times 3}$ is the 3D identify matrix. Note that increasing the order in this way "embeds" lower-order tensor information into higher-order tensors.

In 3D space, the cross product creates a vector orthogonal to two existing vectors. The so-called epsilon tensor (or Levi-Cevita symbol) is underlying the cross product. It is a traceless, antisymmetric tensor of order three. It can be used in a similar way to the delta tensor to contract tensor indices, or to increase the tensor rank. In terms of the epsilon tensor ϵ_{ijk}, the cross product is written as $u^{(1)}_i := \sum_i \epsilon_{ijk} s^{(1)}_j t^{(1)}_k$. Or think of the matrix $\mathbf{T}^2_{\text{anti}} = \begin{pmatrix} 0 & t_3 & -t_2 \\ -t_3 & 0 & t_1 \\ t_2 & -t_1 & ,0 \end{pmatrix}$, which is the result of $t^{(2)}_{jk} := \sum_i \epsilon_{ijk} t^{(1)}_i$ and represents the matrix whose application is a cross product with the vector $t^{(1)}$.

Similar to derivatives, tensor products can be used to successively create higher-order tensors. With an order one tensor as an example, we get

$$t^{(n+1)}_{i_1,\cdots,i_n,i_{n+1}} := t^{(n)}_{i_1,\cdots,i_n} t^{(1)}_{i_{n+1}}. \quad (4)$$

We can imagine that with 3^n, the DOF grows drastically with an increasing order n. For $n = 5$ for example, we already have a DOF of 243. However, in most applications, tensors, like the derivatives and tensors based on outer products as in (4) and (3), have symmetries of the form $t^{(n)}_{i,j,\ldots} = t^{(n)}_{j,i,\ldots}$, or antisymmetries of the form $t^{(n)}_{i,j,\ldots} = -t^{(n)}_{i,j,\ldots}$. So, usually the actual DOF is often by far lower than possible.

For general higher-order Cartesian tensors however, it might be tricky to identify symmetries and redundancies, particularly when it comes to implementation and real data.

Whether a tensor has certain symmetries or not, and thus, can be represented by a fewer number of components, depends on the fact that a tensor of order n may or may not contain vanishing low-order tensor information. Let us consider a second-order tensor $\mathbf{T}^2$ with a maximum of 9 DOF, which can be uniquely decomposed into three components:

$$\mathbf{T}^2 = \underbrace{(tr(\mathbf{T}^2)/3)\mathbf{I}_{3\times 3}}_{\mathbf{T}^2_{\text{tr}}} + \underbrace{(\mathbf{T}^2 - (\mathbf{T}^2)^T)/2}_{\mathbf{T}^2_{\text{anti}}} + \underbrace{(\mathbf{T}^2 + (\mathbf{T}^2)^T)/2 - \mathbf{T}^2_{\text{tr}}}_{\mathbf{T}^2_{\text{sym}}}. \tag{5}$$

The first component $\mathbf{T}^2_{\text{tr}}$ represents the trace of $\mathbf{T}^2$. Since it has only one DOF, it can be represented by a scalar; $\mathbf{T}^2_{\text{anti}}$, an antisymmetric matrix with three DOF, can be written as a first-order tensor lifted by the epsilon tensor, and finally, $\mathbf{T}_{\text{sym}}$ is a traceless symmetric matrix, a second-order tensor with only five DOF (DOF = 6 minus one for the vanishing trace: $tr(\mathbf{T}^2_{\text{sym}}) = 0$). Under a rotation, the elements $\mathbf{T}^2_{\text{tr}}$, $\mathbf{T}^2_{\text{anti}}$ and $\mathbf{T}_{\text{sym}}$ of $\mathbf{T}^2$ do not mix and hence form invariant subspaces. An invariant subspace which cannot be decomposed further into even smaller invariant subspaces is called irreducible. This brings us now to spherical tensors, which are just representations of the tensor within these irreducible subspaces.

2.1 The Irreducible Spherical Tensors

With the decomposition into irreducible subspaces, we can separate independent components by their rotation behavior. This helps to decrease memory requirements due to eliminated redundancies and can lead to more efficiency.

As an example, consider a fully traceless, symmetric Cartesian tensor $\mathbf{T}^2_{\text{sym}}$ of rank 2. Rotating a second-order tensor according to (2) requires $3^2 \times 3 \times 2 = 54$ multiplication. However, we already know that such a tensor has just 5 DOF. The rotation is acting in fully linear manner on these five numbers. So there has to be a 5×5 matrix acting on these five numbers which accomplishes the same task with just $5^2 = 25$ multiplications. And this vectorized representation of the symmetric, traceless matrix is exactly its spherical tensor representation. And in fact, this idea can be generalized to arbitrary order. For higher-order Cartesian tensors, the number of irreducible components becomes large quite quickly [15]: A general order five tensor can already be decomposed into up to 51 irreducible parts; see Table 1 in appendix. The family of these irreducible components is, just like in the order two case, the family of fully symmetric, traceless Cartesian tensors. The DOF of such a tensor of order n is only $2n + 1$ (making it symmetric, reduces the DOF from 3^n to $\binom{3+n-1}{n}$, and then minus $\binom{n}{2}$ (removing all traces) leads to $2n + 1$). Figure 2 illustrates the rotation of the irreducible components in comparison with the reducible representation of an order two tensor. We refer to [6,106] for further details.

Let us now be more explicit: A spherical tensor $\mathbf{a}^j$ of order $j \in \mathbb{N}_0$ is represented by a vector with $2j+1$ complex-valued elements $(a^j_{-j}, \cdots, a^j_j)^T \in \mathbb{C}^{2j+1}$. Note that due to the irreducibility a contraction of tensors does not exist, which is expressed by the fact that there is just a single subindex m. Spherical tensors are, as their Cartesian counterparts, rotation covariant. Spherical tensors of order j are rotated by the so-called Wigner-D rotation matrices $\mathbf{D}^j(g) \in \mathbb{C}^{(2j+1)\times(2j+1)}$ by

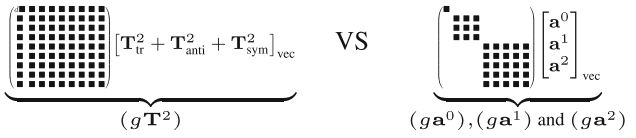

Fig. 2 Cartesian tensors (*left*) can be reduced into irreducible representation which do not mix with each other under rotations (*right*). Irreducible representations foster the development of efficient algorithms

$$(g\mathbf{a}^j) = \mathbf{D}^j(g)\mathbf{a}^j. \tag{7}$$

The Wigner-D rotation matrices (or spherical group representations) are all possible group representations of the rotation group $SO(3)$.

2.2 Clebsch–Gordan Coefficients

The explicit connection between spherical and Cartesian tensors is, despite for the order one tensor, not trivial. Let $\mathbf{S} \in \mathbb{C}^{3\times 3}$ be the unitary transformation that maps the ordinary rotation matrix to the first-order Wigner-D matrix, where

$$\mathbf{D}^1(g) = \mathbf{S}\mathbf{R}(g)\mathbf{S}^\top \text{ and } \mathbf{S} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -\mathbf{i} & 0 \\ 0 & 0 & \sqrt{2} \\ -1 & -\mathbf{i} & 0 \end{pmatrix}. \tag{8}$$

With the matrix $\mathbf{S}$, we have a one-to-one mapping between Cartesian and spherical tensors of order one.

For higher orders, the connection is determined by the so-called Clebsch–Gordan coefficients. They form the basis of the representation theory of 3D rotations. The Clebsch–Gordan coefficients are combinatorial coefficients, which couple two group representations $\mathbf{D}^{j_1}$ and $\mathbf{D}^{j_2}$ to form a new representation. The general law is

$$D^j_{mn} = \sum_{\substack{m_1+m_2=m \\ n_1+n_2=n}} D^{j_1}_{m_1 n_1} D^{j_2}_{m_2 n_2} \langle jm | j_1 m_1, j_2 m_2\rangle \langle jn | j_1 n_1, j_2 n_2\rangle. \tag{9}$$

With $\langle jm \mid j_1 m_1, j_2 m_2\rangle \in \mathbb{R}$, we denote the Clebsch–Gordan coefficients; see Sect. 1 in appendix for further details.

Equations (8) and (9) implicitly define the connection between the Cartesian and spherical representations. For

instance, let $\mathbf{T}^2$ be a second-order, Cartesian tensor. Let further, for convenience, be $\mathbf{T}^s = \mathbf{S}\mathbf{T}^2\mathbf{S}^\top$. Then the components of the corresponding spherical tensors $\mathbf{b}^j$, $j = 0, 1, 2$, are

$$b_m^j = \sum_{m_1+m_2=m} \langle 1m_1, 1m_2|jm\rangle (-1)^{m_2} t^s_{m_1(-m_2)}, \tag{10}$$

where $\mathbf{b}^0$ corresponds to $\mathbf{T}^2_{\text{tr}}$, $\mathbf{b}^1$ to $\mathbf{T}^2_{\text{anti}}$ and $\mathbf{b}^2$ to $\mathbf{T}^2_{\text{sym}}$; see (5). The inverse of this "Cartesian to spherical" transformation is

$$t^s_{m_1m_2} = \sum_{j=0,2} \sum_{m=-j}^{m=j} \langle 1m_1, 1(-m_2)|jm\rangle (-1)^{m_2} b_m^j. \tag{11}$$

In "Cartesian Tensors" section in appendix, we give the explicit expressions of (10) and (11).

2.3 Relation to Fourier Analysis

In contrast to Cartesian tensors, directional information of spherical tensors is rather obscured. However, we can always interpret spherical tensors as expansion coefficients of a spherical Fourier expansion of a square-integrable orientation function $f : S_2 \to \mathbb{C}$, an angular-dependent function on the sphere. In contrast to the tensors themselves, such orientation functions can be indeed interpreted in an intuitive manner. They can be meaningfully visualized in 3D space in tandem with the original image. In STA, the design and interpretation of orientation functions f is, in our opinion, the important objective; the spherical tensors are just the tool to achieve the goals in a numerical manner. The Fourier expansion connecting the tensors with f is given by

$$f(\mathbf{n}) := \sum_{j=0}^{L} \sum_{m=-j}^{m=j} \overline{a_m^j} Y_m^j(\mathbf{n}) = \sum_{j=0}^{L} (\mathbf{a}^j)^T \mathbf{Y}^j(\mathbf{n}), \tag{12}$$

with $L = \infty$. The vector $\mathbf{n} \in S_2$ is a direction (unit) vector in 3D space. The $\mathbf{Y}^j : S_2 \to \mathbb{C}^{2j+1}$ are vectors of $2j + 1$ orthogonal spherical Fourier basis functions Y_m^j of order j, the so-called *spherical harmonics* (see "Spherical Harmonics" section of Appendix for definition). Figure 3 visualizes spherical harmonics up to order 3.

An example of an orientation function is the angular-dependent distribution of gradient directions in a local Gaussian-windowed neighborhood of a voxel. Typical examples are the structure tensor [12,41,52], or with higher-order tensors, spherical histograms of oriented gradients (SHOG) [101]. Figure 4 depicts such local orientation functions. In this example, they represent the local gradient orientation distribution at different image locations in top of the corresponding image.

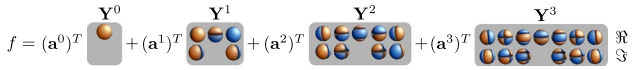

Fig. 3 Spherical harmonic expansion for functions on the sphere $f : S_2 \to \mathbb{C}$. The *upper row* visualizes the real part of the harmonics, the *bottom row* the imaginary parts

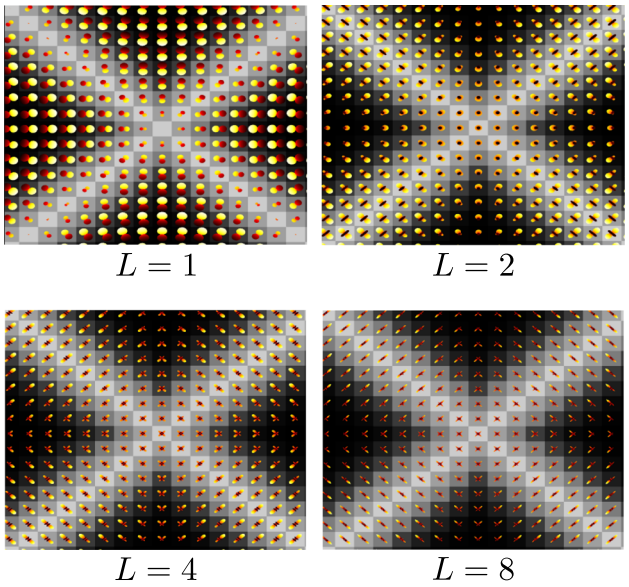

Fig. 4 Orientation functions representing the local distribution of gradient orientations in a Gaussian neighborhood around a crossing. We show local expansions at each voxel, each up to order $L = 8$; see (12). For $L = 2$, the tensors of the expansion are the spherical counterparts of the 3D structure tensor. Only with higher orders, the orientation functions become sharp and can capture the crossing in the center correctly. For better visualization, we have removed the mean from the expansions ($j = 0$)

With (12), the spherical tensors gain some nice properties known from Cartesian Fourier analysis:

Symmetry and DOF Fourier coefficients are encoding the real and imaginary part of complex-valued signals in a separable manner. As a consequence, the Fourier coefficients of real-valued functions $f : S_2 \to \mathbb{R}$ have the characteristic symmetry

$$a^j_{-m} = (-1)^m \overline{a^j_m}. \tag{13}$$

We call the space of such tensors the "real" linear subspace V_j. We call the orthogonal complement $\mathbf{i}V_j$ of V_j the "imaginary" subspace which fulfills $a^j_m = (-1)^{m+1}\overline{a^j_{-m}}$. The space $\mathbf{i}V_j$ corresponds to the expansion coefficients of purely imaginary functions $f : S_2 \to \mathbb{C}$; see "Real and Imaginary Tensor Fields" section in Appendix for details.

As a result, in conformity with irreducible Cartesian tensors, the DOFs of the "real" and "imaginary" spherical tensors of order j shrink from $2(2j + 1)$ to $2j + 1$: the first j complex-valued components and the solely real, or solely

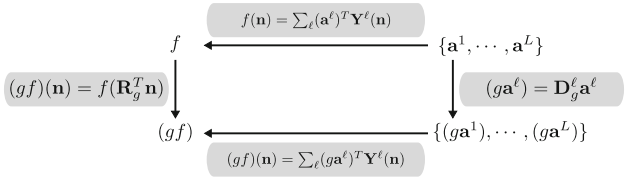

Fig. 5 A rotation of an orientation function in the spatial domain is accomplished with a coordinate transform. The corresponding operation in Fourier domain is a rotation of the expansion coefficients

imaginary, valued center a_0^j, respectively.
Note that for any tensor $\mathbf{a}^j \in V_j$ exists the tensor $\mathbf{i}\mathbf{a}^j \in \mathbf{i}V_j$. Tensors in V_j can be associated with irreducible symmetric Cartesian tensors and tensors in $\mathbf{i}V_j$ with irreducible antisymmetric Cartesian tensors. Hence, an arbitrary irreducible symmetric Cartesian tensor can be expressed by an irreducible antisymmetric Cartesian tensor and vice versa.

Finite Signal Representation Fourier coefficients represent image details in a coarse-to-fine order. We obtain a finite representation of a function f with infinite support by cutting off higher-order frequencies. For this, we set L to a finite number in our applications.
Products The product of two Fourier coefficients is again a Fourier coefficient of an orientation function. We call these products as spherical products. They will be discussed later in this manuscript.
Rotations In the Cartesian Fourier domain, cyclic translations along the Cartesian axis can be accomplished with a rotation (a phase shift) of the corresponding Fourier coefficients. The corresponding transformation for functions on the sphere is the rotation. An orientation function f can be rotated according to the coordinate transform $(gf)(\mathbf{n}) := f(\mathbf{R}(g)^T \mathbf{n})$. Similar to a cyclic translation, f can be rotated in spherical Fourier domain by rotating the Fourier coefficients according to

$$(gf)(\mathbf{n}) := \sum_{j=0}^{L} (\mathbf{D}^j(g)^T \mathbf{a}^j)^T \mathbf{Y}^j(\mathbf{n}). \tag{14}$$

That is, the coefficients of (gf) are the rotated coefficients of f. This relationship is illustrated in Fig. 5.

Axial Symmetric Functions The spherical harmonic expansion of any axial symmetric orientation function of the form $f(\theta, \phi) = f(\phi)$ (z-axis aligned) has only scalar-valued expansion coefficients (all remaining tensor components are vanishing). The expansion simplifies to

$$f(\mathbf{n}) := \sum_{j=0}^{L} \overline{a^j} Y_0^j(\mathbf{n}). \tag{15}$$

In Fig. 21, we show three examples with relevance for image processing applications.

Convolution and Correlation With spherical tensors as Fourier coefficients, spherical convolutions are just products between tensors. The spherical convolution between a spherical function f, with expansion coefficients $\mathbf{a}^j$, and an axial symmetric orientation function f', with scalar-valued expansion coefficients b^j (see 15), is the simple product

$$(f * f')(\mathbf{n}) := \sum_{j=0}^{L} (b^j \mathbf{a}^j)^T \mathbf{Y}^j(\mathbf{n}). \tag{16}$$

3 Spherical Tensor Fields

For images, a tensor typically changes with the location within the image. We call a tensor which changes with respect to the position a *tensor field*. We call the corresponding field of spherical functions (12) *orientation fields*. The creation, processing and interpretation of orientation fields in terms of spherical tensors is the base of all introduced algorithms. In this section, we first explore the rotation of orientation fields, the key property of rotation covariant algorithms. Then we introduce the theoretical background of two major operations of the toolbox: the spherical counterparts of tensor products (4) and derivatives (3). They are used to map images to spherical tensor fields and vice versa.

Orientation fields are functions $f : \mathbb{R}^3 \times S_2 \mapsto \mathbb{C}$ that assign to each point $\mathbf{r} \in \mathbb{R}^3$ in 3-space an orientation function (12). We can write any square-integrable orientation field f according to (12) with respect to its second argument (the direction) as an expansion

$$f(\mathbf{r}, \mathbf{n}) = \sum_{j=0}^{\infty} \mathbf{a}^j(\mathbf{r})^\top \mathbf{Y}^j(\mathbf{n}). \tag{17}$$

The expansion coefficients $\mathbf{a}^j(\mathbf{r})$ are tensors which vary with respect to their location in 3D space.

Any orientation field can be rotated with $(gf)(\mathbf{r}, \mathbf{n}) := f(\mathbf{R}(g)^\top \mathbf{r}, \mathbf{R}(g)^\top \mathbf{n})$ in a "classical" way. The first argument is a coordinate transformation, and the second argument rotates the local spherical function accordingly. With (14), the rotation can be accomplished in the Fourier domain according to

$$(gf)(\mathbf{r}, \mathbf{n}) = \sum_{j=0}^{\infty} \left(\mathbf{D}^j(g)\mathbf{a}^j\left(\mathbf{R}(g)^\top \mathbf{r}\right)\right)^\top \mathbf{Y}^j(\mathbf{n}). \tag{18}$$

That is, if an orientation field f is rotated, the underlying expansion fields $\mathbf{a}^j$ of expansion coefficients $\mathbf{a}^j(\mathbf{r})$ undergo the transformation

$$\mathbf{a}^j(\mathbf{r}) \mapsto \left(\mathbf{D}^j(g)\mathbf{a}^j\left(\mathbf{R}(g)^\top \mathbf{r}\right)\right). \tag{19}$$

We will call any function with this kind of transformation a spherical tensor field of order j.

Definition 1 (*Spherical Tensor Field*) A function $\mathbf{f}^j : \mathbb{R}^3 \mapsto \mathbb{C}^{2j+1}$ is called a spherical tensor field of order j if it transforms with respect to rotations as

$$(g\mathbf{f}^j)(\mathbf{r}) := \mathbf{D}^j(g)\mathbf{f}^j\left(\mathbf{R}(g)^\top \mathbf{r}\right) \tag{20}$$

for all $g \in SO(3)$. The space of all spherical tensor fields of rank j is denoted by $\mathcal{T}_j$.

In this context, it is important noting that an image $I : \mathbb{R}^3 \to \mathbb{R}$ is a spherical tensor field of order 0.

3.1 Spherical Tensor Coupling

With the delta and epsilon tensor, there exist various ways to combine Cartesian tensors. Since Cartesian and spherical tensors can basically express the same quantities, there must exist a counterpart of the products in the spherical tensor domain as well. Thanks to the irreducibility, there neither exists an operator for tensor contraction (there exists no trace), nor a way to represent lower-order tensors in terms of higher-order tensors. As a consequence, there exists only one single inner product-like bilinear form in the spherical tensor domain. We call this operation the *spherical product*.

For example, both a Cartesian and a spherical tensor of order one are irreducible. In this case, there exists a one-to-one relation between the Cartesian and spherical products. Let $\mathbf{U}^1$ and $\mathbf{S}^1$ be two Cartesian tensors of order one. The inner product $t^{(0)} = \sum_{ij} \delta_{ij} u_i^{(1)} s_j^{(1)}$, the cross product $t_i^{(1)} = \sum_{jk} \epsilon_{ijk} u_j^{(1)} s_k^{(1)}$ and the traceless outer product $t_{ij}^{(2)} = u_i^{(1)} s_j^{(1)} - \delta_{ij} t^{(0)}/3$ combine the two tensors and create new irreducible, Cartesian tensors of order zero, one and two.

In the spherical tensor domain, with spherical tensors $\mathbf{u}^1, \mathbf{s}^1 \in V_1$, the corresponding products can be performed with one spherical product $(\mathbf{u}^1 \circ_j \mathbf{s}^1)$, where $j \in 0, 1, 2$. It is defined by a family of bilinear forms:

Definition 2 (*Spherical Products*) For every $j \geq 0$, we define a family of bilinear forms, a tensor product

$$\circ_j : \mathbb{C}^{2j_1+1} \times \mathbb{C}^{2j_2+1} \mapsto \mathbb{C}^{2j+1}, \tag{21}$$

where $j_1, j_2 \in \mathbb{N}$ has to be chosen according to the triangle inequality $|j_1 - j_2| \leq j \leq j_1 + j_2$. The product is defined by

$$(\mathbf{e}_m^j)^\top (\mathbf{v} \circ_j \mathbf{w}) := \sum_{m=m_1+m_2} \langle jm \mid j_1 m_1, j_2 m_2 \rangle v_{m_1} w_{m_2}.$$

With $\mathbf{e}_m^j$, we denote the unit vectors in $\mathbb{C}^{2j+1}$, where $e_{mn}^j = \delta(m-n)$.

The spherical products are, like their Cartesian counterparts, rotation covariant.

Proposition 1 *Let* $\mathbf{v} \in \mathbb{C}^{2j_1+1}$ *and* $\mathbf{w} \in \mathbb{C}^{2j_2+1}$, *then for any* $g \in SO(3)$

$$(\mathbf{D}^{j_1}(g)\mathbf{v}) \circ_j (\mathbf{D}^{j_2}(g)\mathbf{w}) = \mathbf{D}^j(g)(\mathbf{v} \circ_j \mathbf{w}) \tag{22}$$

holds.

That is, it does not matter whether we first rotate the factors or the final product.

The spherical products are nonlinear transformations that can combine whole spherical tensor fields in a point-by-point manner. For example, given two tensor fields $\mathbf{v} \in \mathcal{T}_{j_1}$ and $\mathbf{w} \in \mathcal{T}_{j_2}$ and j chosen such that $|j_1 - j_2| \leq j \leq j_1 + j_2$, then $(\mathbf{v} \circ_j \mathbf{w})(\mathbf{r}) := \mathbf{v}(\mathbf{r}) \circ_j \mathbf{w}(\mathbf{r})$ is in $\mathcal{T}_j$; it is spherical tensor field of order j. Later in this article, products are used in convolutions as well. Convolutions can be used to map local image patches to spherical tensor-valued basis functions, an important step to generate local descriptors.

The spherical product naturally commutes between the "real" and "imaginary" spaces V_j and $\mathbf{i}V_j$. It reflects the parity by its (anti-)symmetry. The toolbox makes uses of these properties in order to save memory and computation time.

Proposition 2 *Let* $\mathbf{v} \in V_{j_1}$ *and* $\mathbf{w} \in V_{j_2}$, *then*

$$\begin{aligned} j + j_1 + j_2 \text{ is even } &\Rightarrow \quad \mathbf{v} \circ_j \mathbf{w} = \mathbf{w} \circ_j \mathbf{v} \in V_j \\ j + j_1 + j_2 \text{ is odd } &\Rightarrow \quad \mathbf{v} \circ_j \mathbf{w} = -(\mathbf{w} \circ_j \mathbf{v}) \in \mathbf{i}V_j, \end{aligned}$$

With the spherical products, we can define a rotation covariant spatial convolution between tensor fields of different orders.

Definition 3 (*Tensor Convolution*) For two tensor fields $\mathbf{v} \in \mathcal{T}_{j_1}$ and $\mathbf{w} \in \mathcal{T}_{j_2}$, the operation

$$(\mathbf{v} \widetilde{\circ}_j \mathbf{w})(\mathbf{x}) := \int_{\mathbb{R}^3} (\mathbf{v}(\mathbf{r}) \circ_j \mathbf{w}(\mathbf{x} - \mathbf{r}))\mathrm{d}\mathbf{r} \tag{23}$$

defines the tensor convolution.

Proposition 3 (Convolution Theorem) *The convolution theorem*

$$(\mathbf{v} \widetilde{\circ}_j \mathbf{w}) = FT^{-1}(FT(\mathbf{v}) \circ_j FT(\mathbf{w})) \tag{24}$$

holds for the tensor convolution. With FT, we denote the ordinary spatial 3D Fourier transform which maps tensor fields to their Fourier counterpart in a component-by-component manner.

3.2 Covariant Feature Extraction

In many cases, we start with a scalar-valued input image, a spherical tensor field of order zero. An initial step is the rotation covariant feature extraction. This procedure maps local image patches to tensor-valued coefficients. In the Cartesian domain, the image derivatives (see 3) often build the basis for covariant or invariant feature extraction. Image derivatives are encoding, with increasing order, local image appearance in a coarse-to-fine manner. Spatially smoothing the derivatives with a kernel, like a Gaussian G_σ, steer the level-of-detail, or the local image patch size. Since the convolution commutes with the differentiation, such a scale-dependent analysis can be realized with only one initial smoothing; i.e.,

$$\underbrace{G_\sigma * \left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_{n+1}}} I\right)}_{\substack{\text{instead of} \\ \text{one convolution per derivative}}} \overset{=}{\rightarrow} \underbrace{\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_{n+1}}} (G_\sigma * I)}_{\text{one convolution in total (fast)}} \tag{25}$$

This is a big advantage, particularly for large images, because image derivatives can be implemented efficiently in terms of finite differences. In our context, (25) is the connection between scalar-valued images and tensor-valued features. In fact, (25) can be interpreted as a projection on Cartesian tensor-valued basis functions $\mathbf{K}^{(n)}$, where

$$\begin{aligned} t^{(n)}_{i_1,\cdots,i_n}(\mathbf{x}) &:= \Big(\underbrace{\left(\frac{\partial}{\partial x_{i_1}} \cdots \frac{\partial}{\partial x_{i_{n+1}}} G_\sigma\right)}_{\text{basis function } k^{(n)}_{i_1,\cdots,i_n}} * I\Big)(\mathbf{x}) \\ &= \left(k^{(n)}_{i_1,\cdots,i_n} * I\right)(\mathbf{x}). \end{aligned} \tag{26}$$

Equation 26 is also called a *sliding dot product*. This projection maps images in a sliding window manner to tensor-valued feature fields. Each tensor at each voxel represents local image features in a rotation covariant manner. The basis functions in the example (26) are known as the Gaussian-windowed Hermite polynomials; see, e.g., [67]. A Gaussian kernel G_σ, however, is not the only possible way to build basis functions with such rotation covariant differential relationship. In fact, any differentiable radial symmetric kernel is a possible candidate.

In this section, we introduce two examples of spherical tensor-based basis functions with differential relationship. This is, on the one hand, the spherical counterpart of the 3D Hermite polynomials. Further, we briefly introduce a spherical tensor-valued Gabor basis.

3.2.1 Spherical Derivatives

The homogeneous polynomials $R^j_m(\mathbf{r}) = r^j Y^j_m(\mathbf{n})$ of order j are called the "solid harmonics"; see Sect. 1 for details. With $r = |\mathbf{r}|$, we denote the distance to the center, and with

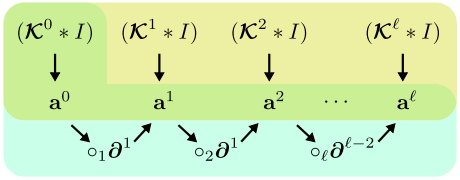

Fig. 6 Finite differences are faster than convolutions based on the Fourier transform. For a kernel with differential, a projection up to order L can be accomplished in two manners: (1) *upper row* projection with $((L+1)(L+2))/2$ convolutions and (2) *bottom row* with one convolution followed by differentiation. The latter one significantly reduces the computation time

$\mathbf{n} = \mathbf{r}/r$ the direction. You may consider them as the bridge between the spherical harmonics, which only exist on the sphere (S_2), and the 3D image space ($\mathbb{R}^3$). With the operator transformation $R^j_m(\nabla)$, we map the Cartesian gradient operator $\nabla = (\partial_x, \partial_y, \partial_z)^T$ to the spherical domain. We define the spherical differential operator $\boldsymbol{\partial}^j := (\partial^j_{-j}, \cdots, \partial^j_j)^T$ by

$$\partial^j_m := R^j_m(\nabla). \tag{27}$$

The operator $\boldsymbol{\partial}^1$, with $\boldsymbol{\partial}^1 = \mathbf{R}^1(\nabla)$, is, as its Cartesian counterpart, a 3D vector, namely $(\frac{1}{\sqrt{2}}(\partial_x - i\,\partial_y), \partial_z, \frac{1}{\sqrt{2}}(-\partial_x - i\,\partial_y))^T$.

We can treat $\boldsymbol{\partial}^j$ like a spherical tensor of order j. Similar to its Cartesian counterpart, higher-order derivative operators can be built out of lower-order ones with

$$\boldsymbol{\partial}^j = \sqrt{\tfrac{j!}{(2j-1)!!}}\, \boldsymbol{\partial}^1 \circ_j (\ldots \circ_3 (\boldsymbol{\partial}^1 \circ_2 \boldsymbol{\partial}^1) \ldots). \tag{28}$$

This property is of particular importance from a computational perspective, because we can compute tensor features in increasing order $j = 0, 1, 2, \cdots$ in a successive manner:

Let $\mathcal{K}^j \in \mathcal{T}_j$ be a spherical tensor-valued basis function with the differential relationship $\mathcal{K}^j = \boldsymbol{\partial}^j \mathcal{K}^0$. Then, the sliding dot product between an image I and the basis functions $\mathcal{K}^j$, $j = \{0, 1, \ldots\}$ can be successively computed, similar to (25) and (26), with one convolution in combination with finite differences:

$$\mathbf{a}^j = (\mathcal{K}^j * I) = \boldsymbol{\partial}^j (\mathcal{K}^0 * I). \tag{29}$$

This relationship is visualized in Fig. 6.

Gauss–Laguerre Gaussian derivatives play an important role in the context of scale space analysis and feature extraction in many image processing tasks; e.g., [39,40,66,67,92,104,118]. Cartesian Gaussian derivatives are called the Hermite polynomials. The spherical counterpart of the 3D Hermite polynomials is the Laguerre polynomials [67]. Gaussian-windowed Laguerre polynomials are, as their Cartesian counterparts, optimal for local smooth processes; see section 5.1.2 in [97].

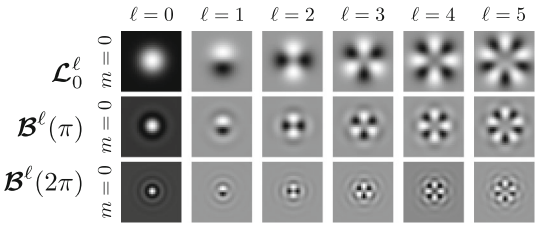

Fig. 7 Images showing the center slice of the real-valued coefficient ($m = 0$) of Gauss–Laguerre and Gabor kernels. For these coefficients, the angular patterns around the Z-axes are 2D circular harmonics of order j

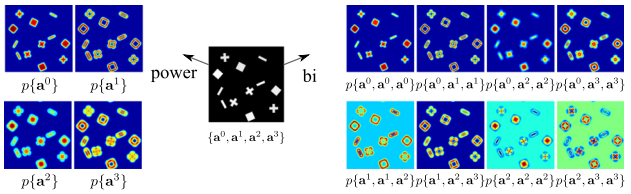

Fig. 8 Power spectrum and the (even) bi-spectrum features of an image. The image has been projected to a Gaussian derivative basis with coefficients up to order three

The Gaussian-windowed Laguerre basis functions $\mathcal{L}_n^j$ of order j are spherical tensor fields of order j. They are defined by

$$\mathcal{L}_n^j(\mathbf{r}) = \sqrt{\frac{j!}{(2j-1)!!}} \frac{(-\sigma^2)^{n+j}}{n! 2^n} \mathbf{R}^j(\mathbf{r}) L_n^{j+\frac{1}{2}}\left(\frac{r^2}{(2\sigma^2)}\right) e^{-\frac{r^2}{2\sigma^2}}, \quad (30)$$

with the differential relationship

$$\mathcal{L}_n^j = \Delta^n \partial^j \mathcal{L}_0^0; \quad (31)$$

see [103] for details. Note that $\mathcal{L}_0^0$ is the 3D Gaussian function. With Δ, we denote the Laplace operator, and with L_n^α the associated Laguerre polynomial [1].

Gabor Functions The functions

$$\mathcal{B}^j(\mathbf{r}, k) = (-k)^j \sqrt{\frac{j!}{(2j-1)!!}} \mathbf{Y}^j(\mathbf{n}) J_j(kr) \quad (32)$$

are the natural radial basis functions appearing in the spherical expansion of the plane wave; see (70) in appendix. With J_j, we denote the spherical Bessel functions [1]. With

$$\mathcal{B}^j(k) = \partial^j J_0(kr) \quad (33)$$

we have a differential relationship similar to the Gauss–Laguerre functions. In our applications, we use a Gaussian-windowed version $\mathcal{B}^j(k) G_\sigma$ with local support, a Gabor wavelet. See [103] for the exact expression and differential formulation in terms of Gabor wavelets. Figure 7 shows examples for $\mathcal{B}^j(k)$ and $\mathcal{L}_n^j$.

3.3 Rotation Invariant Features

We call any zero-order tensor $\mathbf{a}^0$ *rotation invariant* in the sense that with

$$(g\mathbf{a}^0) = \mathbf{D}^0(g)\mathbf{a}^0 = \mathbf{a}^0, \quad (34)$$

the transformation is the identity transform. The corresponding tensor fields $\mathbf{f} \in \mathcal{T}_0$ transform like ordinary 3D images; i.e., they rotate according to $(g\mathbf{f})(\mathbf{x}) := \mathbf{f}(\mathbf{U}(g)\mathbf{x})$. Hence, the quantities, the tensors themselves, are not altered. We call this property local rotation invariance. All 3D biological images with scalar-valued intensities are fields with locally rotation invariant zero-order tensors.

In a 3D biological or medical feature detection tasks, it is often desired to detect objects, or specific structures, in a rotation invariant manner. With STA, it is possible to create a set of zero-order tensor fields. Each of them contains discriminative, mutually exclusive features of local image structures in a rotation invariant manner. The idea is that even if the location or the orientation of an object varies, only the location of the set of corresponding invariant features undergoes a coordinate transform—the feature itself stays constant.

A set of locally rotation invariant features is a signature of local image structures. Any state-of-the-art classifier can be used in this context in a sliding window approach to identify the objects at any location and in any orientation. In Fig. 8, we show two examples of rotation invariant features based on the spherical power spectrum and the spherical bi-spectrum of an image.

Given Fourier coefficients, the power spectrum might be the most commonly used invariant feature. The power spectrum represents the energy distribution, or power, with respect to the frequencies, in our case the tensor orders.

Definition 4 (*Power Spectrum*) The power of an order $\mathbf{a}^j$ tensor can be computed with the spherical product

$$p\{\mathbf{a}^j\} = (\mathbf{a}^j \circ_0 \mathbf{a}^j). \quad (35)$$

The power spectrum is often sufficient when objects clearly differ in symmetry. It reduces the characteristics of a function to a small number of coefficients. However, this might sometimes not be enough. For instance, although the functions

$$f(\mathbf{n}) = \sum_{j=0}^{\infty} (\mathbf{a}^j)^\top \mathbf{Y}^j(\mathbf{n}) \text{ and}$$
$$f'(\mathbf{n}) = \sum_{j=0}^{\infty} (\mathbf{b}^j)^\top \mathbf{Y}^j(\mathbf{n}); \mathbf{b}^j := \begin{cases} \mathbf{a}^j & \text{if } j \neq 2 \\ \mathbf{D}(g)^j \mathbf{a}^j & \text{else} \end{cases} \quad (36)$$

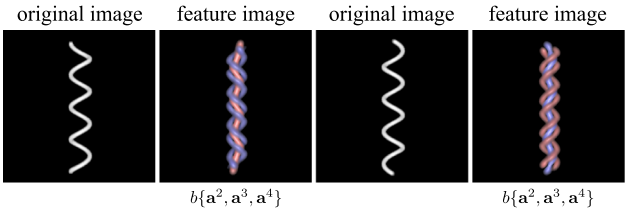

Fig. 9 Odd bi-spectrum features can discriminate point reflections. In the feature images, positive contributions are visualized in *red*, negative contributions in *blue*

differ in the orientation of the order two tensor coefficient, their power spectra are identical, see Proposition 1. This is because the power spectrum does not preserve any information about connections between the relative directional information of the coefficients. This is a disadvantage over the bi-spectrum.

Definition 5 (*Bi-Spectrum*) Let $\mathbf{a}^j \in \mathbb{C}^{2j+1}$, $j = \{0, 1, 2, \ldots\}$ be the spherical expansion coefficients of some spherical function. The bi-spectrum, we refer to [61] for details, is formed by all possible spherical products of order 3 that return a scalar (a tensor of rank 0):

$$b\{\mathbf{a}^{j_1}, \mathbf{a}^{j_2}, \mathbf{a}^{j_3}\} = (\mathbf{a}^{j_1} \circ_{j_3} \mathbf{a}^{j_2}) \circ_0 \mathbf{a}^{j_3}. \tag{37}$$

The bi-spectrum has a variety of nice properties. For example, contrarily to the power spectrum, it can discriminate point reflections: The odd products ($j_1 + j_2 + j_3$ is odd) change their signs if a the underlying object undergoes a point reflection (or any other orthogonal transformation with negative determinant), see Fig. 9 for an example.

The spherical bi-spectrum is a specific case of the bi-spectrum over the rotation group $SO(3)$ (The functions over all three Euler angles (θ, ϕ, ψ); we only deal with spherical functions over the two angles (θ, ϕ)). For $SO(3)$, the bi-spectrum is complete [61,62]. This is a powerful property. It means the object can be fully recovered from the bi-spectrum (up to the rotation). However, in case of $SO(3)$, it is a rather large object, and many possible redundancies exist. Restricting on spherical functions, the bi-spectrum becomes a rather simple object and redundancies can be avoided easily by following some specific coupling rules for triple products, which we discuss in the next subsection. Further details regarding its properties are found in [61].

3.3.1 Symmetries in Triple Products

Computing all permutations of products between three spherical tensors, as in the bi-spectrum, leads to a set of linearly dependent tensors. Fortunately, thanks to the broad usage of spherical tensors in the angular momentum theory, the symmetric relationships of angular momentum states have been studied exhaustedly in detail [15,88]. In our case, the relationship between three states is important, see (98) in appendix.

Considering the symmetries can resolve those linear dependencies. In fact, the bi-spectrum is independent of the ordering of the arguments j_1, j_2, j_3, that is, there are only $\binom{L}{3}$ independent bi-spectral invariants, if the spherical signal is band-limited by L. The following corollary summarizes the most important cases. Theorem 1 covering the general case together with the proofs is found in appendix.

Corollary 1 (Associativity of Tensor Products) *For the triple tensor products, there exist coupling rules that are one-to-one, see* [97]. *These rules are*:

(Bi-spectrum) If $|j_1 - j_2| \leq j_3 \leq j_1 + j_2$, *then*

$$\begin{aligned}&\left(\left(\mathbf{u}^{j_1} \circ_{j_3} \mathbf{v}^{j_2}\right) \circ_0 \mathbf{w}^{j_3}\right) = \left(\left(\mathbf{w}^{j_3} \circ_{j_2} \mathbf{u}^{j_1}\right) \circ_0 \mathbf{v}^{j_2}\right)\\ &= \left(\left(\mathbf{v}^{j_2} \circ_{j_1} \mathbf{w}^{j_3}\right) \circ_0 \mathbf{u}^{j_1}\right).\end{aligned} \tag{38}$$

(Upper Bound) If $j_1, j_2, j_3 \in \mathbb{N}$, *then*

$$\begin{aligned}&\left(\left(\mathbf{u}^{j_1} \circ_{(j_1+j_2)} \mathbf{v}^{j_2}\right) \circ_{(j_1+j_2+j_3)} \mathbf{w}^{j_3}\right)\\ &= \left(\left(\mathbf{u}^{j_1} \circ_{(j_1+j_3)} \mathbf{w}^{j_3}\right) \circ_{(j_1+j_2+j_3)} \mathbf{v}^{j_2}\right)\\ &= \left(\left(\mathbf{v}^{j_2} \circ_{(j_2+j_3)} \mathbf{w}^{j_3}\right) \circ_{(j_1+j_2+j_3)} \mathbf{u}^{j_1}\right).\end{aligned} \tag{39}$$

(Lower Bound) If additionally $(j_3 - j_2 - j_1) \geq 0$, *then*

$$\begin{aligned}&\left(\left(\mathbf{w}^{j_3} \circ_{(j_3-j_1)} \mathbf{u}^{j_1}\right) \circ_{(j_3-j_2-j_1)} \mathbf{v}^{j_2}\right)\\ &= \left(\left(\mathbf{w}^{j_3} \circ_{(j_3-j_2)} \mathbf{v}^{j_2}\right) \circ_{(j_3-j_2-j_1)} \mathbf{u}^{j_1}\right)\\ &= \left(\left(\mathbf{u}^{j_1} \circ_{(j_1+j_2)} \mathbf{v}^{j_2}\right) \circ_{(j_3-j_2-j_1)} \mathbf{w}^{j_3}\right).\end{aligned} \tag{40}$$

3.4 Steerable Voting

In Sect. 3.2, we mapped local image patches to spherical tensor-valued basis functions $\mathcal{K}^j$. A result was a set of expansion coefficients $\{\mathbf{a}^j\}$. In this section, we introduce the dual operation, which "renders" patches into a scalar-valued image. Applications can be steerable filters, where an elongated filter kernel is rendered into a target image in consistency with tubular structures in a source image; see, e.g., [50]. Or, as we will see as an example in the application section, to create object specific saliency maps, we call them *voting maps*, for the generic detection of 3D objects.

In Refs. [71,81,91,101], this idea was used in the spirit of a generalized Hough transform [8]. Let $\{\mathbf{a}^j\}$, with $j = \{0, 1 \cdots\}$, be the set of expansion fields of a local voting function. Then, the process of "rendering" the votes into an

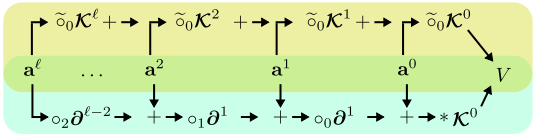

Fig. 10 Finite differences are faster than convolutions based on the Fourier transform. For a kernel with differential relationship, steerable voting can be accomplished in two manners: (1) *upper row* many expensive projections with convolutions (2) *bottom row* Differentiation followed by one single convolution. The latter one significantly reduces the computation time

image can be accomplished with a convolution-like operation:

$$V(\mathbf{r}) = \int_{\mathbb{R}^3} \sum_{j=0}^{\infty} (\mathbf{a}^j(\mathbf{x}))^T \mathcal{K}^j(\mathbf{r}-\mathbf{x}) d\mathbf{x}, \tag{41}$$

where $V \in \mathcal{T}_0$ is a scalar-valued image. This operation is rather computationally expensive. However, if $\mathcal{K}^j$ is a kernel with differential relationship, see 3.2, then (41) can be computed in the following manner:

$$V = \sum_{j=0}^{L} (\mathbf{a}^j \widetilde{\circ}_0 \mathcal{K}^j) = \mathcal{K}^0 * \sum_{j=0}^{L} (\boldsymbol{\partial}^j \circ_0 \mathbf{a}^j); \tag{42}$$

see Sect. (23) for the definition of the tensor convolution. By computing the derivatives in a top-down manner, we only need L first-order spherical derivatives in combination with an ordinary, scalar-valued convolution to compute the final result. This makes both feature computation and voting computational efficient. Figure 10 illustrates the work flow.

3.5 Diffusion on $\mathbb{R}^3 \times S_2$ with STA

Image enhancement/restoration schemes are often based on diffusion/convection schemes. In this section, we shortly describe the generators of diffusion and convection on $\mathbb{R}^3 \times S^2$, and we show how they can be implemented in terms of spherical tensor algebra.

Most of the implementations solving $\mathbb{R}^3 \times S^2$-diffusion equations [10,28,29,84] rely on an equiareal discretization of the two-sphere S_2. Indeed, there are implementations [29] that use spherical harmonics as an intermediate S_2-interpolation scheme, but due to their design they cannot benefit from the advantages of the spherical harmonic representation, like the efficient computations of S_2-convolutions, and the closedness under rotations. Below, we introduce a common differential operators acting on functions $\mathbb{R}^3 \times S_2 \mapsto \mathbb{C}$ in terms of spherical harmonics. In this way, we can benefit from the advantages of the harmonic representations. The discretization of the S_2 is avoided, and we are able to implement diffusion on $\mathbb{R}^3 \times S_2$ with reasonable memory consumption. In general, we are interested in solving or propagating a partial differential equation of the form

$$\partial_t f(\mathbf{r}, \mathbf{n}, t) = Hf(\mathbf{r}, \mathbf{n}, t)\,, \tag{43}$$

where f is a time-dependent orientation field, and H a linear differential operator in $\mathbf{n}$ and $\mathbf{r}$. In [29], it is shown that if H generates a diffusion/convection, then it is a quadratic form in $\mathbf{n}^\top \nabla$ (convection and anisotropic diffusion), and linear in $\Delta = \partial_x^2 + \partial_y^2 + \partial_z^2$ and $\mathcal{J}^2$, which denotes the Laplace–Beltrami operator on the two spheres. Our goal is to understand the action of the generator H, if the field f is written in terms of a spherical harmonics expansion, see (17). The spherical tensor fields $\mathbf{a}^j$, the expansion coefficients, are obtained by the projections $\mathbf{a}^j(\mathbf{r}) = \frac{1}{2j+1}\langle \mathbf{Y}^j, f(\mathbf{r})\rangle$. Hence, we are interested in matrix elements $\hat{H}^{jm}_{j'm'} = \langle Y^{j'}_{m'}, HY^j_m\rangle$ of H in spherical harmonic representation such that the propagating equation can be written in the form

$$\partial_t a^j_m(\mathbf{r}, t) = \sum_{j=0}^{\infty} \sum_{m=-j}^{m=j} \hat{H}^{jm}_{j'm'} a^{j'}_{m'}(\mathbf{r}, t) \tag{44}$$

where $\hat{H}^{jm}_{j'm'}$ is a differential operator in $\mathbf{r}$, but purely algebraic in the orientation coordinate. The spherical Laplace operator is well known in this representation. It is defined by

$$\left\langle Y^j_m, \mathcal{J}^2 f\right\rangle = -j(j+1)a^j_m. \tag{45}$$

For the spatial Laplacian, the result is $\langle Y^j_m, \Delta f\rangle = \Delta a^j_m$. For the directed convection $\mathbf{n}\cdot\nabla$ and the directed diffusion $(\mathbf{n}\cdot\nabla)^2$ generator, the result is not that trivial. In [87], we give the general result for $SE(3)$ diffusion, but mention here the more simple case for $\mathbb{R}^3 \times S_2$. For the convection generator, one finds

$$\begin{aligned}\left\langle Y^j_m, (\mathbf{n}\cdot\nabla) f\right\rangle &= \sum_{\substack{j'=-1,1\\ m=m'+q}} \frac{2j'+1}{2j+1} \langle jm|j'm', 1q\rangle \langle j0|j'0, 10\rangle\, \partial^1_q a^{j'}_{n'} \\ &= \sum_{j'=-1,1} \frac{2j'+1}{2j+1} \langle j0|j'0, 10\rangle \boldsymbol{\partial}^1 \circ_j \mathbf{a}^{j'}\,. \end{aligned} \tag{46}$$

On the other hand, the diffusion generator takes the form

$$\begin{aligned}\left\langle Y^j_m, (\mathbf{n}\cdot\nabla)^2 f\right\rangle &= \frac{\Delta}{3} + \frac{2}{3} \\ &\times \sum_{\substack{j'=-2,0,2\\ m=m'+q}} \frac{2j'+1}{2j+1} \langle jm|j'm', 2q\rangle \langle j0|j'0, 20\rangle\, \partial^2_q a^{j'}_{n'} \\ &= \frac{\Delta}{3} + \frac{2}{3} \sum_{j'=-2,0,2} \frac{2j'+1}{2j+1} \langle j0|j'0, 20\rangle \boldsymbol{\partial}^2 \circ_j \mathbf{a}^{j'}. \end{aligned} \tag{47}$$

The diffusion generator is an ideal candidate for regularizing inverse problems where line-like structure is of interest. In [84], this regularizer is called 'fiber continuity' (FC).

4 The STA-Toolbox

The STA-toolbox provides a set of operations and procedures which ease the handling of spherical tensors. This includes spherical tensor products, derivatives, as well as a Fourier transformation respecting symmetries of spherical tensors. It further provides higher-level API functions for feature extraction, object detection or anisotropic filtering. For performance reasons, the toolbox is mostly written in C and C++, but provides a high-level API in MATLAB. The toolbox has been successfully tested on a 64 bit Windows system and a 64 bit Linux system. On Linux, multi-threaded tensor operations are enabled by default. Tensor operations are also available on the GPU, written in CUDA.

A stafield class encapsulates the data structure of orientation fields. Most of the functions of the STA-toolbox have a simple API in MATLAB, or in its open-source alternative OCTAVE. In this section, we introduce the basic functionality of the toolbox using the MATLAB interface. References to the C++ and C interface can be found online on the project webpage.

4.1 Spherical Tensor Fields

The basis of all calculations is a data container that stores the expansion coefficients of an orientation field $f : \mathbb{R}^3 \times S_2 \to \mathbb{C}$; see (17) for the definition. The expansion coefficients are a (band-) limited number of spherical tensor fields $\{\mathbf{a}^0, \mathbf{a}^1, \cdots, \mathbf{a}^L\}$, where $\mathbf{a}^j \in \mathcal{T}_j$.

We call this data container a *stafield*. It is a multi-dimensional array with attributes describing the properties of the orientation field in terms of its expansion coefficients. The data are stored in a five-dimensional, real-valued array of size $2 \times N \times X \times Y \times Z$. Figure 11 illustrates the memory alignment of the data. The first dimension is always two. It represents the real and imaginary part of the tensor field. The second dimension represents all tensor field components a^j_m; we will give details about the storage order below. The third, fourth and fifth dimensions are the image dimensions.

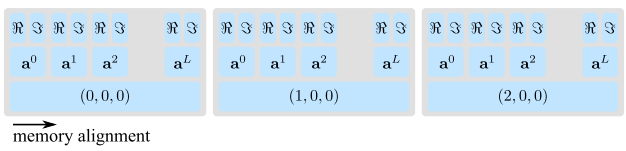

Fig. 11 Representation of stafields in memory. It is an interleaved array of the real and imaginary values of the tensor coefficients

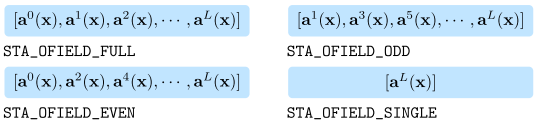

Fig. 12 The toolbox provides four different types of orientation field representation. The full field contains all coefficients up to a specified order, while the single field contains only a single spherical tensor field

A tensor field is associated with the four attributes: *storage*, *L*, *type* and *element_size*:

1. The *element_size* is a three-dimensional vector which defines the extents of a voxel (e.g., in micrometers). The default is $(1, 1, 1)^T$. The voxel size is considered by both the tensor derivatives and convolution kernel sizes. We recommend to use this attribute to account for an anisotropic image resolution.
2. The attribute *L* defines the upper limit of the expansion.
3. A stafield may contain only a single spherical tensor field, or only a certain subset of coefficients with odd or even order.
   The toolbox distinguishes between four different field *type*s: "single," "odd," "even" and "full." "Single" is the standard type and defines an orientation field that is defined by a single spherical tensor field of order *L*. With "full," the array contains *L*+1 tensor fields, ranging from order 0 up to order *L*. "Odd" and "even" are basically "full" tensor fields; however, in order to save memory, all even, or odd tensors, respectively, are omitted in the array. Figure 12 illustrates these four types.
4. In most of the applications, spherical tensor fields contain only real tensors, see Sect. 2.3. In this case, the toolbox stores only the lower $l+1$ parts $(a^l_{-l}, \ldots, a^l_0)^T$ of a tensor field of order l and sets the *storage* attributes to "STA_FIELD_STORAGE_R". Otherwise the full tensor is stored ("STA_FIELD_STORAGE_C").

A stafield in MATLAB is represented by the `stafieldStruct` MATLAB structure.

The `stafieldStruct` constructor can turn any real-valued image into an orientation field. Below an example for a real-valued image called "img."

```
>> size(img)
ans =
     80     80     80

>> sfield=stafieldStruct(img)
sfield =
     storage: 'STA_FIELD_STORAGE_R'
           L: 0
        type: 'STA_OFIELD_SINGLE'
        data: [5-D single]
     element_size: [1 1 1]
```

Basic tensor operations, which may alter the tensor order, like products and derivatives, can only be applied to stafields of the type "single," that is, single spherical tensor fields. In order to apply such operations to all kinds of stafields, the toolbox provides an interface to access the single components, the spherical tensor fields, of a stafields container. With the `stafieldStruct`, we can access single stafields from full stafields. For example,

```
% extracting the order 3 field from the
% full stafield ifield
ofield=stafieldStruct(ifield,3),
% ofield is a stafield of type
% `single' of order 3
```

extracts the order three field component from `ifield`, and we overwrite a single component in a full stafield with a given single field with

```
% suppose ifield is an l-th order
% single stafield.
% overwriting the l-th order component
% in the full stafield ofield with
% ifield
ofield=stafieldStruct(ofield, ifield)
```

Note that in C and C++, these operations work without the need for making expensive memory copies.

The stafield constructor can also initialize specific tensor fields for feature extraction. In addition to the Gauss–Laguerre and Gabor kernels $\mathcal{L}_n^j$ and $\mathcal{B}^j(k)$, see Sect. 3.2, it supports Gaussian kernels [81] and Gaussian smoothed spheres [101]. For example,

```
kernel=stafieldStruct('gauss',...
                ,[128,128,128],5)
```

initializes a stafield of order 0 using a Gaussian function with a standard deviation of 5. By default, a kernel is centered with respect to $(0, 0, 0)^T$ so that it can directly be used in a tensor convolution. Optional parameters change the tensor field order, define the *storage* type, or set the *element_size*.

The `stafieldStruct` offers many further ways to construct or initialize empty tensor fields. Typing `help stafieldStruct` lists detailed information about further constructors.

4.1.1 Cartesian to Spherical Tensors and Vice Versa

In most applications, it is sufficient to either work with Cartesian tensors OR with spherical tensors. Mixing both worlds is, in general, from an implementation and computation point of view, in our opinion, not recommended. Remember, table 1 gives some examples about the large number of the different possible irreducible components of a higher-order Cartesian tensor. On the other hand, tensors of order one or two are playing an important role in many existing algorithms so that such a forward and backward transformation is, indeed, demanded. According to (5), a second-order tensor can always be decomposed into a unique triple of irreducible order two, one and zero tensors.

The toolbox provides functions to connect first- and second-order Cartesian tensor fields with their spherical counterparts and vice versa. The toolbox provides both the forward and backward transformation. The Cartesian-to-spherical transformation can be accomplished with the `sta_c2s`. For the back transformation, there exists the `sta_s2c` function. The two functions `sta_Grad` and `sta_Hessian` are serving as a reference for storage convention of Cartesian tensor fields of order one and two, respectively. The following example shows how to transform a Cartesian gradient and Hessian field into their spherical counterparts.

```
% gradient field to tensor field
field=sta_c2s(sta_Grad(img));
% and the back trafo
T1=sta_s2c(field);

% Hessian field to tensor field
field=sta_c2s(sta_Hessian(img));
% and the back trafo
T2=sta_s2c(field);
```

4.2 HARDI to Spherical Tensors and Vice Versa

Diffusion-weighted magnetic resonance imaging (MRI) can noninvasively visualize the fibrous structure of the human brain white matter [58]. Based on the directional dependency of water diffusivity in fibrous tissue, it is possible to directly acquire orientation information. The high angular resolution diffusion imaging (HARDI) acquisition scheme, where more than 60 diffusion directions per voxels are acquired, allows to estimate the orientation distributions of local fiber bundles. Such a measurement is essential for in vivo fiber tracking [85]. The measurement is an orientation field, which is often represented in terms of spherical tensor expansion coefficients for further processing steps like feature detection or deconvolution. In its raw form, it is an orientation field represented by a four-dimensional array whose forth dimension corresponds to the diffusion direction. The toolbox provides the functions

```
% converting HARDI signal to
% an even stafield
[ofield,b0avg]=...
    sta_hardi2stafield(data,b_dir);
% b0avg is the b0 signal

% converts a stafield into a
% HARDI signal
data=sta_stafield2shardi(ofield,...
      b0avg,b_dir);
```

which map from a discrete HARDI signal to an even[3] stafield and vice versa.

4.3 Spherical Tensor Operations

The toolbox provides the following basic tensor operations which are acting on the stafieldStruct container:

Operation	Matlab function	Interpretation
Spherical products	`sta_prod`	$(\mathbf{f} \circ_j \mathbf{g})$, $\mathbf{f} \in \mathcal{T}_j, \mathbf{g} \in \mathcal{T}_k$
Spherical derivatives	`sta_deriv`	$(\boldsymbol{\partial}^n \circ_{n+m} \mathbf{f})$, $n = \{-2, 1, 0, 1, 2\}$, $\mathbf{f} \in \mathcal{T}_m$
Laplace	`sta_lap`	$\Delta\mathbf{f}$, $\mathbf{f} \in \mathcal{T}_m$
Tensor FFT	`sta_fft, sta_ifft`	$\widetilde{\mathbf{f}} = \mathrm{FT}(\mathbf{f})$, $\mathbf{f} = \mathrm{FT}^{-1}(\widetilde{\mathbf{f}}); \mathbf{f}, \widetilde{\mathbf{f}} \in \mathcal{T}_m$
Multiplication	`sta_mult`	$\alpha\mathbf{f}$, $\alpha \in \mathbb{C}$, $\mathbf{f} \in \mathcal{T}_m$

Spherical Products Given two stafields `ifield1` and `ifield2` of the type "single," the spherical product can be computed according to

```
ofield3=sta_prod(ifield1,ifield2,L),
```

where L is the tensor rank of the output ofield3. Optional parameters can switch between a standard and a normalized tensor product, or can weight the product by a given scalar.

Spherical Derivatives Given a stafield `ifield1` of the type "single," the spherical derivative operator can be applied with

```
ofield2=sta_deriv(ifield1,n),
```

where n can be either -2 (two times down-derivative), -1 (one time down-derivative), 0 (a curl like operation, which is not changing the tensor order), 1 (one time up-derivative) and 2 (two dimes up-derivatives).

Laplace Operator The Laplace operator can be applied to any type of stafield. The output is a field with the same attributes like the input field.

Tensor FFT The FFT operator can be applied to any type of stafield.

Multiplication The multiplication operator is a wrapper that allows (complex valued) scalar multiplications with stafields. While for real-valued factors $\alpha \in \mathbb{R}$

```
field.data=field.data*alpha;
```

is a valid solution, the C-style of complex numbers requires the usage of

```
field=sta_mult(field,alpha_r+1i
*alpha_i);
```

for complex-valued factors.

[3] The signal is symmetric.

4.3.1 An Introductory Example

A fundamental step in many applications involving STA is the covariant feature extraction, see Sect. 3.2. For steerable filters, for instance, as well as for the computation of rotation invariant features, an image is locally expanded in terms of a Gauss–Laguerre or Gabor basis.

That is, computing a stafield `ofield` $:= (\mathbf{a}^0, \mathbf{a}^1, \cdots, \mathbf{a}^j)$, with $\mathbf{a}^j := (\mathcal{K}^j * I)$,where $\mathcal{K}^j$ has the differential relationship $\mathcal{K}^j := \boldsymbol{\partial}^j \mathcal{K}^j$. The projection can be completely performed in MATLAB using the basic STA-toolbox API. For instance,

```
% img to tensor field
ifield=stafieldStruct(img)

% convolution kernel
kernel=stafieldStruct('gauss',...
          ifield.shape,5);

L=3;
% initializing a full orientation
% field of order L
ofield=stafieldStruct(ifield.shape,
L,...
  'STA_OFIELD_FULL',...
  'STA_FIELD_STORAGE_R','single');

% convolution between image and kernel
ifield=sta_ifft(sta_prod(...
    sta_fft(ifield),sta_fft(kernel),
    0,...
    {'alpha',1/prod(ifield.shape)}));

% defining the 0 order field
% of the full field
ofield=stafieldStruct(ofield,ifield);

% iteratively computing the higher
    order
% components
for l=1:L
ofield=stafieldStruct(ofield,...
  sta_deriv(stafieldStruct(ofield,
   l-1),1));
end;
```

computes a full stafield containing all the expansion coefficients up to order 3.

Particularly for large images of several gigabyte, the bottleneck is rather memory allocation and de-allocation than CPU performance. The same operations can be performed using the `sta_steerFilt` function. The `sta_steerFilt` has been designed in a memory friendly way directly in C++. The example above can be shortened to

```
ifield=stafieldStruct(img);
L=3;
ofield=sta_steerFilt(ifield,...
  {'kname','gauss','kparams',5,'BW',
    L,...
    'type','STA_OFIELD_FULL'});
```

The `sta_steerFilt` function can be used to compute Gaussian derivatives, Gauss–Laguerre expansion coefficients as well as Gauss–Bessel coefficients. There are further optimized functions for specific applications, which will be introduced in the application section.

It is worth mentioning that for MATLAB users, there exists a convenient stafield class which mimics the C++ interface. The stafield class implements all tensor operations as member functions. For instance, `ofield` $= \boldsymbol{\partial}^1(G_3 * I)$ can be written as

```
ofield=stafield(img).fft...
.prod(...
     stafield('gauss',size(img),3).fft,
     ... 0)...
.ifft(1/numel(img)).deriv(1);
```

Unfortunately, this interface is currently not supported in Octave.

The toolbox can be extended with further optimized functions using the C++ or C interface of the stafield class. The C++ stafield class can be used in a similar manner than the MATLAB stafield class. The C interface is similar to the stafieldStruct interface in MATLAB and Octave. Helper functions for the exchange of data between MATLAB and C are provided as well. The C++ stafield class also provides member functions to transfer stafields from CPU to GPU and vice versa.

5 Applications

In the following, we give several application examples which make use of STA. The focus lies on applications with value for the biological and medical image analysis. As already outlined in introduction, STA bridges the gap between low-level feature detection of ridge and plane-like structures to higher-order structures. In this section, we show examples of covariant higher-order feature extraction. We further show how to compute rotation invariant descriptor maps for invariant feature detection. We also introduce an example of a trainable polynomial filter. We demonstrate how to utilize polynomial filters to detect complicated 3D image structures. Another example uses a model-based deconvolution approach regularized by the $S_2 \times \mathbb{R}^3$-diffusion generator to enhance and detect tubular structures in 3D. We finally use STA to estimate the neural fiber orientation distributions from magnetic resonance images (MRI).

At the end of each subsection, you may find a simple skeleton that serves as a copy&paste example which runs even on lightweight computer systems. The skeletons together with the data can be found in the *skel* toolbox directory in the repository as well.

5.1 Covariant Feature Extraction

The STA-toolbox provides two categories of covariant features: projection-/convolution-like features and distribution-/histogram-like features. The first group of features is introduced in Sect. 3.2. Details for the latter features will be given in this section. All features are computed densely on the whole volume. Dense feature maps occupy a lot of memory, and their computation is expensive. However, the features we propose here can be computed efficiently with the local spherical derivative and spherical product operators.

5.1.1 Projection Features

Projections on spherical tensor-valued kernels with differential relationship can be realized efficiently via spherical derivatives, see Sect. 3.2.1. For the spherical derivatives, by default, the toolbox uses finite differences of second-order accuracy. They offer the best trade-off regarding computation time and accuracy. For higher-order harmonics, we provide higher-order approximations as well, see Fig. 15 for a comparison regarding time and accuracy.

The function `sta_steerFilt` is an efficient implementation for projections of the kind $\mathbf{a}^j := (\mathcal{K}^j * I)$, where $\mathcal{K}^j$ is a kernel with differential relationship according to Sect. 3.2.1.

Gaussian Derivatives Simple spherical Gaussian derivatives can be computed with

```
ofield=sta_steerFilt(stafieldStruct
(img),...
  {'kernel','gauss','kparam',sigma,...
   'type','STA_OFIELD_FULL','L',L});
```

The parameter L is the cutoff of the expansion, see Sect. 2.3. The parameter *sigma* determines the width of the Gaussian. For further details regarding spherical Gaussian derivatives, see [81].

Gauss–Laguerre The Gauss–Laguerre coefficients can be computed with

```
ofield=sta_steerFilt(stafieldStruct
(img),...
  {'kernel','gaussLaguerre',...
   'kparam',[sigma,n],...
   'type','STA_OFIELD_FULL','L',L});
```

The parameter pair $[sigma, n]$ determines the width of the Gaussian and the polynomial order of the radial Laguerre polynomial, respectively.

Gabor Functions For the Gabor coefficients, we just change the kernel parameter. For instance,

```
ofield=sta_steerFilt(ifield,...
  {'kernel','gaussBessel',...
   'kparam',[sigma,k,s],
   'type','STA_OFIELD_FULL','L',L});
```

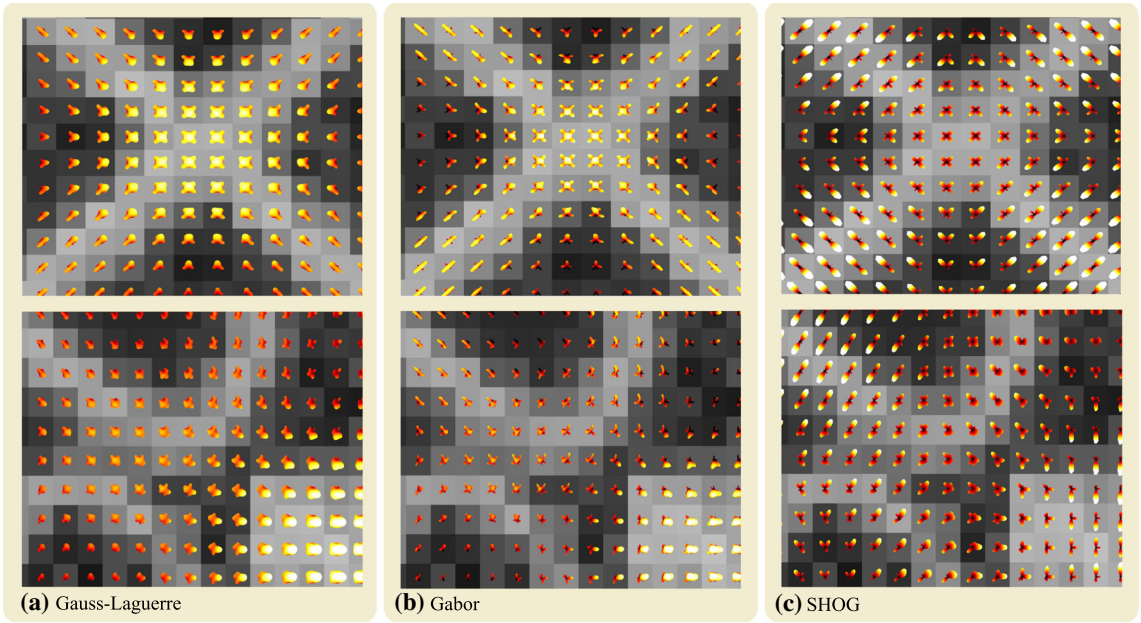

Fig. 13 In these images, the orientation fields are representations of local image features, like edges, crossings, curvature. The orientation functions have been visualized on top of the source image. From the *left* to the *right*, we show orientation fields based on the expansion coefficients of three different types of features: **a** Gauss–Laguerre, **b** Gabor and **c** SHOG

computes the projection of a Gabor with a radial frequency k. The parameter $sigma$ determines the scale of the kernel. The third parameter, s, determines the Gaussian window size of the kernel with respect to the wave frequency k. For details, see [97].

5.1.2 Histogram Features

Histogram of oriented gradients features (HOG) [26]) are a widely used family of discriminative image patch descriptors. HOG features represent the gradient orientation distribution in a local image patch. In their original form, they are not rotation covariant. An extension to rotation covariant 3D representations, which we call SHOG, has been proposed in [70,101].

SHOG (Spherical HOG) Let $\mathbf{g} = \nabla I$ be the gradient image of an image I. Let further $G_s : \mathbb{R}^3 \to \mathbb{R}$ be a radial symmetric window function (here a Gaussian). Then, the orientation functions of the orientation field

$$f\{I\}(\mathbf{x}, \mathbf{n}) := \int_{\mathbb{R}^3} \|\mathbf{g}(\mathbf{r})\|^\gamma \delta^2_{\mathbf{n}}(\hat{\mathbf{g}}(\mathbf{r})) G_s(\mathbf{x} - \mathbf{r}) d\mathbf{r} \tag{48}$$

are representing the local occurrence of gradient orientations, weighted by their magnitude. The function $\hat{\mathbf{g}} := \mathbf{g}/\|\mathbf{g}\|$ is the normalized gradient direction field of $\mathbf{g}$. With the Dirac delta function $\delta^2_{\mathbf{n}} : S_2 \to \mathbb{R}$ (see (69) in appendix), we mask out all gradients in the gradient image $\mathbf{g}$ with a different orientation than $\mathbf{n}$. With the parameter $\gamma \leq 1$, the influence of large outliers in the gradient magnitudes can be reduced, and edges with low signal can be enhanced.

The fields of spherical tensor expansion coefficient of $f\{I\}$ (see (17)) can be computed with

$$\mathbf{a}^j = \left(\left(\|\mathbf{g}(\mathbf{r})\|^\gamma (\mathbf{Y}^j(\hat{\mathbf{g}})) \right) * G_s \right); \tag{49}$$

for details see (5.6.1) in [97].

The function $f\{I\}$ maps gradient orientation information onto a spherical function in a nonlinear fashion. Hence, a fast projection in terms of derivatives, like for the kernels with differential relationship, is not possible. However, based on the fact that higher-order spherical harmonics can be computed in terms of lower order harmonics, see (64) in appendix, we have developed a recursive algorithm for an efficient mapping, see Fig. 14.

The corresponding toolbox function is the `sta_shog` function. A function call is

```
ofield=sta_shog(stafieldStruct(img),...
    {'kernel','gauss',...
     'kparam',s,'BW',L,'gamma',0.8});
```

This example computes the expansion coefficients up to order L of local SHOG features in a Gaussian window of width s.

Copy&Paste Example The following example shows how to computes covariant features and how to visualize the corresponding orientation field. Figure 13 shows examples of the visualization of orientation fields of various features.

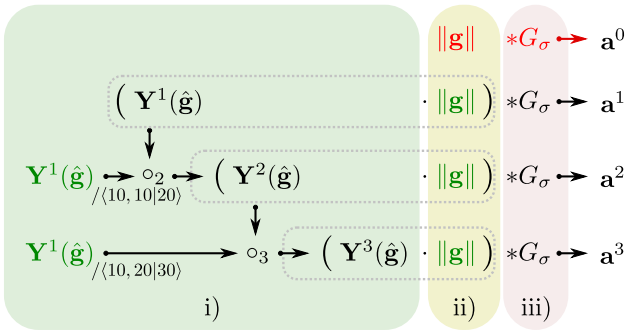

Fig. 14 Based on the gradient field $\mathbf{g} : \mathbb{R}^3 \to \mathbb{R}^3$ of an image, the expansion coefficients of local gradient orientation distributions can be computed in a recursive manner. In this sketch, $\hat{\mathbf{g}} := \frac{\mathbf{g}}{\|\mathbf{g}\|}$ is the gradient direction field of $\mathbf{g}$; also note that $\|\mathbf{g}\|\mathbf{Y}^1(\hat{\mathbf{g}})$ coincides with the spherical derivative $(\boldsymbol{\partial} \circ_1 I)$ which can be computed with the toolbox using the spherical derivative operator. (i) Higher-order representations of the orientation fields, (ii) local weighting of the higher-order direction fields with the gradient magnitudes, and (iii) aggregation of contribution from neighboring voxels using a radial symmetric window function $w : \mathbb{R}^3 \to \mathbb{R}$

```
%% covariant descriptor field
load('skel_data.mat','dataset');

img_id=1;
img=double(dataset{img_id}.img);
img=img+0.25*randn(size(img));

sigma=2.5;
ofield=sta_steerFilt(stafieldStruct(img),...
       {'kernel','gauss','kparam',[sigma],...
       'type','STA_OFIELD_FULL','L',4});

% show the corresponding orientation field
sta_glyph(ofield,'init_view',3,'gamma',-0.9);
```

5.2 Rotation Invariant Descriptor Fields

In volumetric images, objects like cells, neurites, blood vessels, parts of tissue or bones may appear in any orientation. In contrast to 2D, where a planar rotation has only one degree of freedom, a 3D rotation has three of them. As a consequence, operations that are feasible in 2D, like sliding matched filters, are expensive in 3D so that invariant features play an important role in 3D. Rotation invariance can either be achieved by normalization or by group integration. In fact, both operations can be reduced to spherical tensor products [97]. As mentioned, the STA-toolbox provides functionality to densely compute spherical tensor products. With tensor products, the toolbox can compute angular power- and bi-spectra from local image patches, which we have introduced in Sect. 3.3.

Descriptor fields of spherical spectra have been used in applications for cell detection [93,103], but also for fetal brain and thorax analysis in MRI [59,60] and feature-based tissue classification in HARDI [94,99]. An n-dimensional descriptor image is a mapping $\mathbf{d} : \mathbb{R}^3 \to \mathbb{R}^n$, where the value transform under rotation is the identity transform, namely

$$(g\mathbf{d})(\mathbf{x}) := \mathbf{d}(\mathbf{U}(g)^T\mathbf{x}) \,. \tag{50}$$

The n-dimensional vectors $\mathbf{d}(\mathbf{x})$ in $\mathbf{d}$ are called local (rotation invariant) image descriptors. Given an image descriptor $\mathbf{d}(\mathbf{x})$, any kind of classifier is able to perform a segmentation or classification tasks in a rotation invariant manner. The toolbox provides the optimized function `sta_invrts` to either compute descriptor fields based on the power spectra,

$$d_j := p\{\mathbf{a}^j\}, \tag{51}$$

or the bi-spectra

$$d_k := b\left\{\mathbf{a}^{j_1}, \mathbf{a}^{j_2}, \mathbf{a}^{j_3}\right\}, \tag{52}$$

from the expansion coefficients of an orientation field. This function automatically takes care of product associativities, see Sect. 3.3.1.

HARDI Power-Spectrum Descriptors Given a high angular resolution diffusion image (HARDI) [112]. Such a measurement is essential for in vivo fiber tracking [85], see Sect. 4.2. For fiber tracking, it is important to identify voxels which corresponds to white matter or gray matter tissue. A typical approach is to co-register with mask images that have been registered to a T1 image. A problem is that typical HARDI images have a low spatial resolution and are suffering from strong noise and distortion artifacts. This makes the co-registration error-prune. The HARDI signal is a symmetric orientation field with an equidistant sampling of the angular space. The symmetry implies that only the even coefficients of a spherical harmonic expansion contribute to the expansion. For HARDI signals, the expansion (17) simplifies to

$$f(\mathbf{r}, \mathbf{n}) = \sum_{j=0}^{\infty} \mathbf{a}^{2j}(\mathbf{r})^\top \mathbf{Y}^{2j}(\mathbf{n}). \tag{53}$$

Invariant features based on a tensor decomposition of the HARDI signal have been explored quite frequently in the literature; see for instance [46,48,51,72,95]. In [94], a learning-based approach was introduced. The power spectrum of the spherical harmonic expansion coefficients of the HARDI signal has been used to classify tissue into brain matter, white matter and the background. With the STA-toolbox, these features can be computed in only two steps: first the conversion of the HARDI array representation into an `stafieldStruct`, and then computing the invariants.

```
% converting HARDI signal to an even
% orientation field (default L=4)
ofield=sta_hardi2stafield(data,b_dir);

% computing the local image descriptor
% image using power spectrum features
d=sta_invrts(ofield, 'power2',true);
```

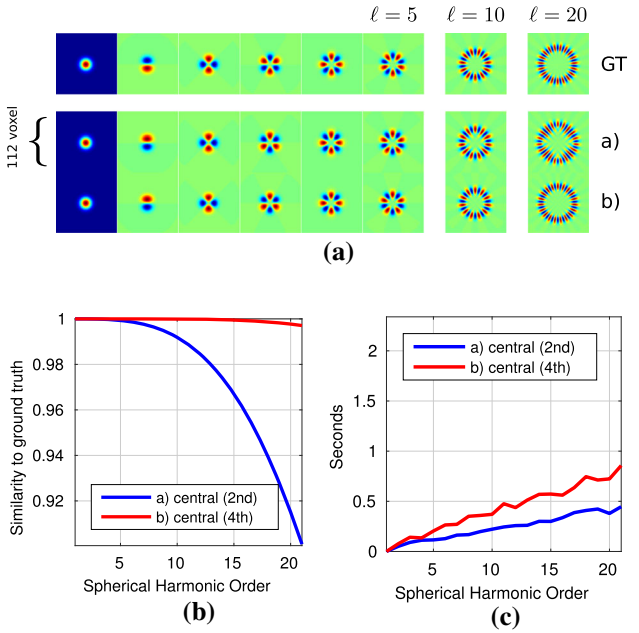

Fig. 15 Accuracy and computation time of tensor derivatives. For further details regarding the experiment, we refer to section 4.2.1 in [97]. **a** Qualitative comparison. (a) Second-order and (b) fourth-order approximation. GT is the explicitly computed harmonic. [(The center slice of a 112^3 volume is shown.) **b** Accuracy (Normalized cross-correlation with explicitly computed harmonics (GT).] (c) Computation speed (Intel Quad Core i5-2400 CPU @ 3.1 GHz)

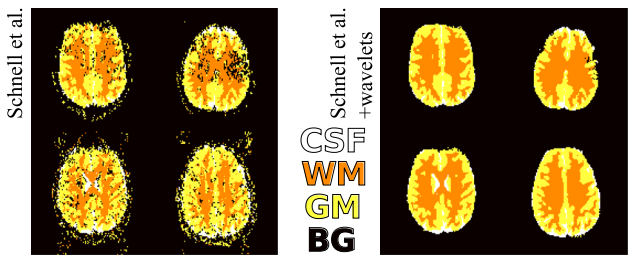

Fig. 16 A voxel classification of a HARDI signal into four classes: background (BG), cerebrospinal fluid (CSF) and brain *gray/white* matter tissue (GM/WM). We show results for four different test images. On the left, we show the classification results based on the features proposed by Schnell et. al. On the *right*, the results of an extension with local derivatives

On the left side of Fig. 16, we show the label predictions of a random forest [14] on a test set. The forest has been trained on five images using the power spectrum of the HARDI signal.

As shown in Fig. 16, this approach leads to a noisy, discontinued segmentation of the HARDI images. HARDI features can easily be enriched with local wavelet features which improves the results. The idea is that not only the raw signal itself, but also rotation invariant representations of local neighborhood descriptors are included into the feature vector. For this example, we use the function `sta_wavelet`. The function `sta_wavelet` can project, in contrast to `sta_steerFilt`, not only images, but also higher-order spherical tensor fields $\mathbf{f}^i \in \mathcal{T}_i$ to expansion coefficients of an orientation function. In our example here, we map the coefficients from (53) to new coefficients $\{\mathbf{b}^{j(0)}, \cdots, \mathbf{b}^{j(i)}\}$, where

$$\mathbf{b}^{\ell(j)} := \left(\partial^{|\ell-j|} \circ_\ell (\mathbf{a}^j * G_\sigma)\right). \tag{54}$$

We then form power spectrum-based descriptor images from those coefficients. For details regarding this projection, see section 5.1.3.1 in [97].

In the following example, we extended the HARDI features with the orientation fields of these wavelet features in three different scales ($\sigma = 0.5$, 1 and 6).

```
% converting HARDI signal to an even
% orientation field (default L=4)
ofield=sta_hardi2stafield(data,b_dir);

% defining orientation field attributes.
% note that we include the raw signal itself
% (it is already an orientation field)
ofield_params={...
  @(x)(x),...
  @(x)sta_wavelet(stafieldStruct(x,0),...
      {'kname','gauss','kparams',0.5}),...
  @(x)sta_wavelet(stafieldStruct(x,2),...
      {'kname','gauss','kparams',0.5}),...
  @(x)sta_wavelet(stafieldStruct(x,4),...
      {'kname','gauss','kparams',0.5}),...
  @(x)sta_wavelet(stafieldStruct(x,0),...
      {'kname','gauss','kparams',1.0}),...
  @(x)sta_wavelet(stafieldStruct(x,2),...
      {'kname','gauss','kparams',1.0}),...
  @(x)sta_wavelet(stafieldStruct(x,4),...
      {'kname','gauss','kparams',1.0}),...
  @(x)sta_wavelet(stafieldStruct(x,0),...
      {'kname','gauss','kparams',6.0}),...
  @(x)sta_wavelet(stafieldStruct(x,2),...
      {'kname','gauss','kparams',6.0}),...
  @(x)sta_wavelet(stafieldStruct(x,4),...
      {'kname','gauss','kparams',6.0}),...
    };

% computing orientation fields
for o=1:numel(ofield_params),
  ofields{o}=ofield_params{o}
(ofield_HARDI);
end;

% computing the local image descriptor
% image using power spectrum features
d=sta_invrts(ofields, 'power2',true);
```

Figure 17 shows the feature images. Thanks to this additional information, particularly the gray matter could be identified more precisely, as shown on the right-hand side of Fig. 16.

Copy&Paste Example The following example shows how to extract rotation covariant features, create rotation invariant features to form a descriptor image and finally to train and apply a classifier Γ to predict labels in an unseen image. The procedure is illustrated in Fig. 18.

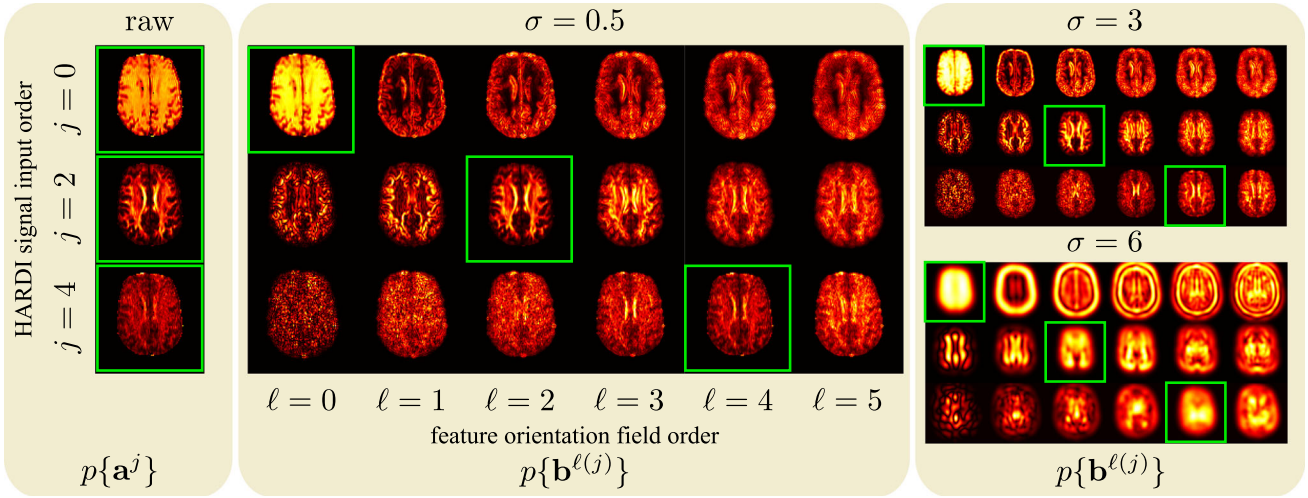

Fig. 17 With only two lines of code, we can compute the band-limited power spectrum $p\{\mathbf{a}^j\}$ features of the HARDI signal (*left column*). This can be used for tissue classification [94]. We can improve the classification performance by including voxel neighborhood information into the descriptor image. Therefore, we compute the power-spectrum features of neighborhood descriptor fields $\mathbf{b}^{\ell(j)}$ in three scales $\sigma = \{0.5, 3, 6\}$. The *green rectangle* indicates the power spectrum feature images with $j = \ell$. Note that we masked out the air (*black*) for a clearer visualization

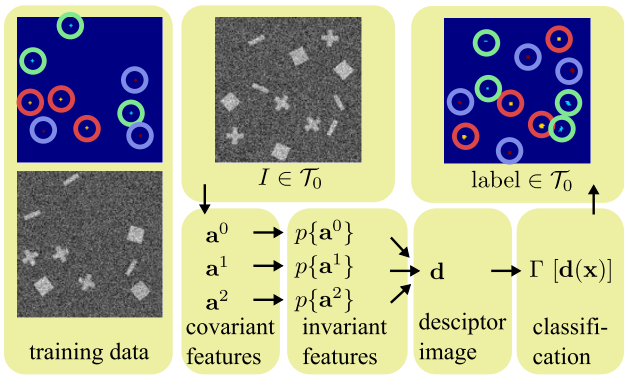

Fig. 18 Pipeline for a label prediction task based on rotation invariant descriptor fields. (See the copy & paste example of Sect. 5.2 for an implementation example). On the *left* the pair of image and labels for training. On the *right*-hand side: the prediction of an unseen image

```
%% rotation invariant object detection
load('skel_data.mat','dataset');
img_train=1;img_test=2;
d={};img={};

for img_id=1:2
 img{img_id}=double(dataset{img_id}.img);
 img{img_id}=img{img_id}+0.25*randn(size(img{img_id}));

 ofields={};
 k=[pi,pi/2]; sigma=4;
 %% computing covariant features (Gabor) for two radial
frequencies
 for a=1:2
     ofields{a}=sta_steerFilt(stafieldStruct(img{img_id}),...
        {'kernel','gaussBessel','kparam',[sigma,k(a),1],...
        'type','STA_OFIELD_FULL','L',5});
 end;
 %% computing the power spectrum
 d{img_id}=sqrt(sta_invrts(ofields, 'power2',true));
end;
%% TRAINING
%%1) preparing the ground truth
gt=double(dataset{img_train}.GT);

imwrite(gray2rgb(squeeze(img{img_train}(:,:,ceil(end/2))),gray)...
        ,'512_train.png');
imwrite(gray2rgb(squeeze(gt(:,:,ceil(end/2))),jet),'512_train_gr
.png');

%%2) sample some negative examples
```

```
neg_examples=(rand(size(gt))>0.999).*bwdist(gt>0)>3;
neg_examples=neg_examples | ...
      (rand(size(gt))>0.8).*(bwdist(gt>0)>1).*(bwdist(gt>0)<6);
train_labels=[find(neg_examples(:));find(gt(:)>0)];

%%3) train a classifiers
myClassifier = ClassificationKNN.fit(d{img_train}
(:,train_labels)',...
       gt(train_labels)','NumNeighbors',3);

%% TESTING
%%3) predict lables on test image
predict=reshape(myClassifier.predict(d{img_test}(:,:)'),...
      size(img{img_test}));
imwrite(gray2rgb(squeeze(img{img_test}(:,:,ceil(end/2))),gray)...
      ,'512_test.png');
imwrite(gray2rgb(squeeze(predict(:,:,ceil(end/2))),jet),...
      '512_test_pred.png');
```

5.3 Steerable Voting

The STA-toolbox implements the voting, dual to `sta_steerFilt`, by `sta_voteFilt` (see Sect. 3.4).

There exists a high-level API, which combines the projection and the voting into a trainable filter. It implements filters presented in [81,91,100–102]. Trainable filters are a family of steerable filters. A common characteristic of these filters is that the shape of the voting kernel can be adapted to a specific 3D object detection task in an image context-dependent manner. The input is an image $I \in \mathcal{T}_0$, the output a saliency map $V \in \mathcal{T}_0$ for the presence of 3D objects or shapes.

For example, a training step, where we have given pairs of images and binary label images of the form {img1,label_img1}. With the training data, we can train a harmonic filter [81] with

```
model=sta_gfilter_train(...
     {img1,img2},...
%two training images
     {label_img1,label_img2},...
%two label images
     5,...
```

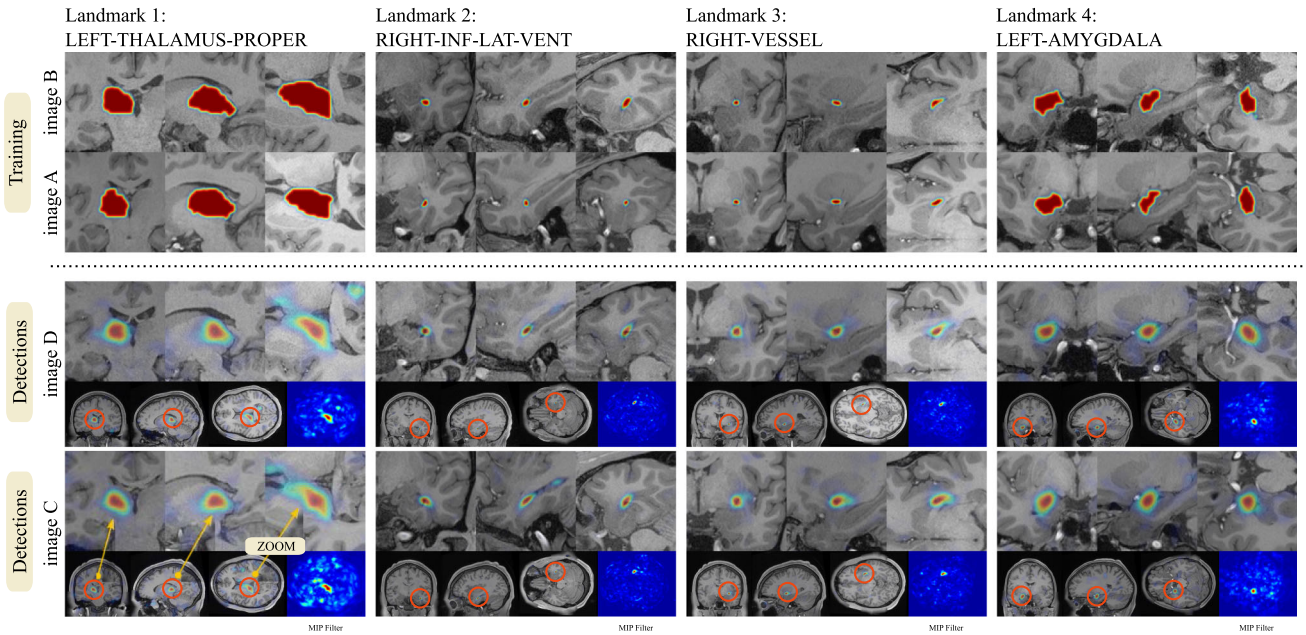

Fig. 19 A filter has been trained on two images A and B to detect four types of landmarks in T1-weighted images of human brains. *Each column* corresponds to one landmark. The *top row* shows the binary-valued label images, which mark the landmarks, on top of the input training images. The landmarks are shown in X, Y and Z-slices, centered at the landmark center. The *two rows* at the *bottom* show results on unseen test images C and D. We show the voting map of the filter in *top* of the input image. In *blue*, we show the maximum intensity projections of the voting image V

```
 %L=5
     {'gauss',3},...
%use wavelet features
     {'gauss',1.5,3,6},...
 %use three voting functions
     'options_combo',...
     {'o2_options_power',[1,1,5,0,5],...
     %second order products
      'o3_options_power',[1,0,1,1,5,
       0,5]},...
      %odd third order products
     'featurefunc',@sta_wavelet_inorm);
 %contrast normalized wavelets
```

We have trained the filter on a landmark detection task in T1 weighted MR images; similar to [100], where they used HARDI data. For the filter, we used second-order covariant features as well as odd third-order covariant features. Due to the reflection symmetry in the brain, odd third-order features are necessary to distinguish the right and left hemisphere. We applied the filter to unseen images with

```
H=sta_gfilter_apply(img,model);
```

The results are shown in Fig. 19.

Copy&Paste Example The following example shows how to train and apply a voting filter. The work flow of the voting is illustrated in Fig. 20.

```
%% steerable voting with trainable filters
load('skel_data.mat','dataset');

img_train=1;img_test=2;
d={};img={};
```

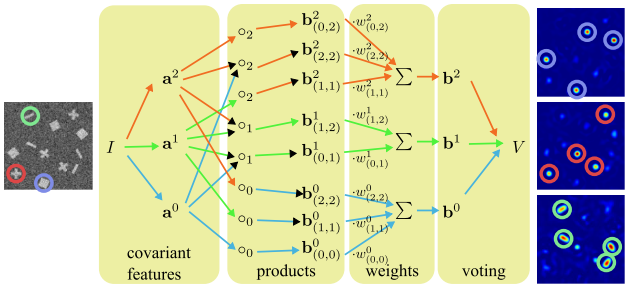

Fig. 20 The trainable voting filter is some kind of trainable steerable filter. The coefficients $\mathbf{a}^j$ are representing local image features of an image I. The coefficients $\mathbf{b}^j$ determine the shape of local voting kernels that are rendered into a saliency map V in an image content dependent manner. The weights $w^j_{(j_1,j_2)}$ are the free parameters. In a training step, the weights are determined so that local patches determine the shape of V. The processing step in the *middle*, here the products, has to be a rotation covariant, in order to get good results, also nonlinear, step. For the copy&paste example, we trained three sets of parameters for the three different kinds of objects

```
for img_id=1:2
 img{img_id}=double(dataset{img_id}.img);
 img{img_id}=img{img_id}+0.25*randn(size(img{img_id}));
end;

for c=1:3;
 gt=(dataset{img_train}.GT)==c;
 % training a model
 model=sta_gfilter_train(...
   {img{img_train}},{gt},5,{'gauss',[1.5],[2.5]},{'gauss',5},...
   'featurefunc',@sta_wavelet,'options_combo',...
   {'o2_options_power',[1,1,5,1,5]});
 imwrite(gray2rgb(squeeze(img{img_train}(:,:,ceil(end/2)))),
    gray),...
    ['513_train',num2str(c),'.png']);
 imwrite(gray2rgb(squeeze(gt(:,:,ceil(end/2))),jet),...
    ['513_train',num2str(c),'gt.png']);
```

```
% predictions on unseen image
predict=max(sta_gfilter_apply(img{img_test},model),0);

imwrite(gray2rgb(squeeze(img{img_test}(:,:,ceil(end/2))),
    gray),...
    ['513_test',num2str(c),'.png']);
imwrite(gray2rgb(squeeze(predict(:,:,ceil(end/2))),jet),...
    ['513_test',num2str(c),'pred.png']);
end;
```

5.4 Steerable Deconvolution

Usually, for low-level feature detection, a bank of steerable filters is correlated with an image to obtain an orientation field [3,5,42]. The orientation field represents the evidence for the presence of simple structures like lines, ridges or edges in a position- and direction-dependent manner. The filter kernels are typically axial symmetric kernels. In Fig. 21, we have visualized examples of such kernels. The STA-toolbox implements the `sta_steerFilt` command, which can correlate an image with a variety of predefined kernels. In the STA-toolbox, a kernel is represented in terms of spherical tensor-valued basis functions $\mathcal{K}^j$. As a consequence, the correlation coincides with projections of the form $\mathbf{a}^j = (\mathcal{K}^j * I)$; it is an extraction of covariant features (Sects. 5.1.1 and 3.2).

In many low-level feature detection tasks, simple structures like tubular-shaped blood vessel or neurites are corrupted by noise and artifacts. In such cases, a simple linear filter may provide a noisy and corrupted orientation field as well. Steerable deconvolution provides a way to obtain a smooth, edge-preserving orientation field from corrupted image data. For the toolbox, we have extended the existing steerable deconvolution in [86] to 3D.

Steerable filters and steerable deconvolution are using the same kind of filter bank. The difference between a steerable filter and the steerable deconvolution is the way we obtain the underlying orientation field. In case of the steerable filter, it is a projection (Sect. 3.2). In the case of steerable deconvolution, we use the dual operation, the voting (Sect. 3.4). Let V_f be a voting field $V_f(\mathbf{r}) = \int_{\mathbb{R}^3} \sum_{j=0}^{\infty} (\mathbf{a}^j(\mathbf{x}))^T \mathcal{K}^j(\mathbf{r}-\mathbf{x}) d\mathbf{x}$, as defined in (41). Then, in steerable deconvolution, we search the coefficients $\mathbf{a}^j$ of the orientation field f (see (17)) which minimize the energy function

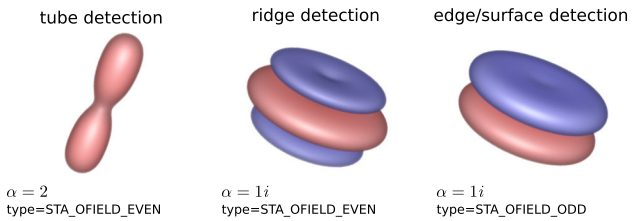

Fig. 21 The toolbox provides a large set of predefined kernels for image filtering and low-level feature detection. This includes kernels for tube detection, 3D ridge detection and surface detection. The *bottom row* shows the corresponding parameters for the steerable filter function `sta_steerFilt`

$$J_{\text{vote}}(f) := \int_{\mathbb{R}^3} |V_f(\mathbf{r}) - I(\mathbf{r})|^2 \mathbf{d}r. \tag{55}$$

That is, steerable deconvolution interprets low-level feature detection as an inverse problem: A steerable filter bank maps a hidden 'orientation' function f onto an observed intensity image I. The corresponding optimization is a deconvolution problem. As the problem is highly underdetermined, a regularization is necessary. The spherical diffusion generator, which we introduced in Sect. 3.5, is an ideal candidate to do so. That is, we want to find an orientation function f such that

$$J(f) = J_{\text{vote}}(f) + \lambda \iint_{\mathbb{R}^3 \times S_2} (\mathbf{n} \cdot \nabla f(\mathbf{r}, \mathbf{n}))^2 \ \mathbf{d}\mathbf{r}\mathbf{d}\mathbf{n} \tag{56}$$

is minimized; an advantage of this 'inverse' approach is that spurious correlation can be suppressed. For example, in a line detection task, crossings can be cleanly resolved, and contributions at intermediate angles are avoided. The STA-toolbox provides a simple API to apply steerable deconvolution to lines, edges and planes.

In Figs. 24 and 25, we show quantitative and qualitative results for both steerable filters and the steerable deconvolution approach.

Figure 22 shows an example obtained for neurites in a drosophila fly brain. The results are obtained via

```
% load some data into an image img
y = sta_steerdeconv(img,'lambda',0.1,
'L',6);
```

which uses the even part of a Gauss–Laguerre expansion $\mathcal{L}_n^j$ restricted to $n = 0$ up to order $L = 6$. One can nicely see how with increasing expansion order L the resulting orientation functions, particularly at crossings, become sharper.

Copy&Paste Example This example loads and performs the steerable deconvolution on the image from Fig. 1.

```
%% steerable deconvolution

load('skel_data2.mat','img3D');
img3D=exp(-bwdist(img3D).^2/2);
img3D=img3D+0.25*randn(size(img3D));

imwrite(gray2rgb(squeeze(img3D(:,:,ceil(end/2))),gray),'
    514_raw.png');

%% steerable deconvolution
res = sta_steerdeconv(stafieldStruct(img3D),'alpha',2,
'maxit',20);

%% local maxmima detection (is a direction)
m = getLocalMinMax(res.data,res.L,1,true,res.type,res
        .storage);
%% maginute
m = squeeze(sqrt(sum(m.^2,1)));

imwrite(gray2rgb(squeeze(m(:,:,ceil(end/2))),gray),'514_filt
    .png');
```

5.5 High Angular Resolution Diffusion Imaging and Spherical Deconvolution

In diffusion-weighted magnetic resonance imaging (MRI) exist high-quality schemes, like high angular resolution dif-

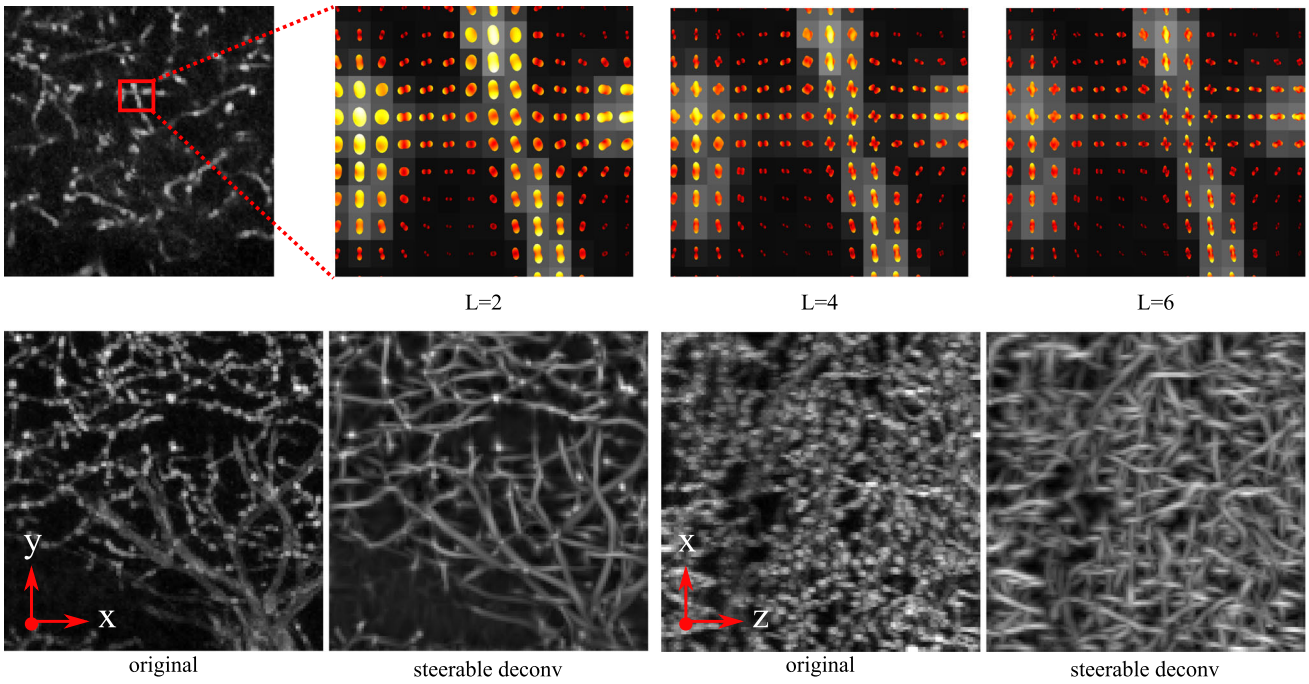

Fig. 22 Varying intensities, gaps, blob-like structures and crossings of presynaptic sites in an image of dopaminergic neurites in a drosophila brain make the tracing of neurites difficult. The orientation field from higher-order steerable deconvolution smoothes the neurites in an anisotropic manner and provides the information to resolve the crossings. The images show the estimation of the orientation field of the neurites using steerable deconvolution. *Upper row* the orientation field based on a band-limited expansion up to order two, four and six. *Bottom row* A maximum intensity projection of the orientation field's maximum responds along z-direction of an $128 \times 128 \times 40$ fraction of a 3D micrograph (*left*), and a projection along y of an $128 \times 40 \times 128$ fraction (*right*)

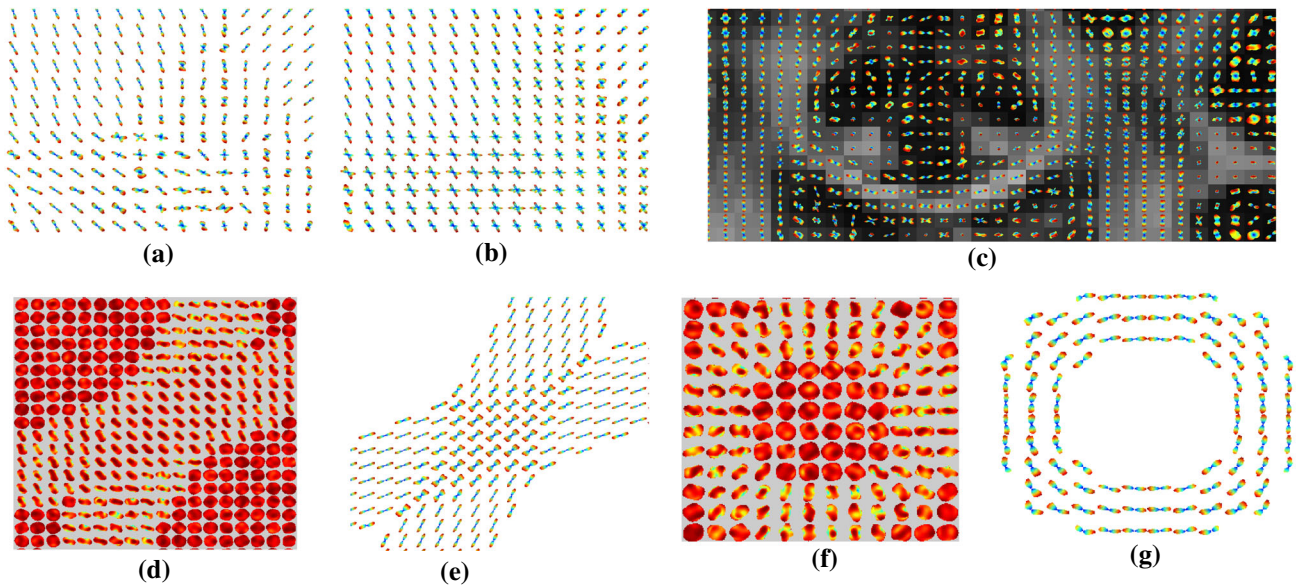

Fig. 23 Application of spatially regularized spherical deconvolution: **a**, **b** examples on a numerical phantom, on the one hand without regularization (a) and with regularization (**b**) for low-quality data (DTI). **d**–**g** Other, more simple numerical phantoms for a crossing with the raw data in **d**, **f** and deconvolution results regularized by FC (**e**) and AFC (**g**). Finally, in **c** a in-vivo example is given. The *background* shows the so-called fractional anisotropy (normalized standard deviation of eigenvalues of the diffusion tensor) in *gray scale*

fusion imaging (HARDI), where more than 60 directions per voxels are acquired, which allow to estimate so-called fiber orientation distributions (FOD). There also exist low-quality datasets, suitable for clinical routine, that allow just the estimation of a diffusion tensor (DTI). To obtain the FOD, a physical diffusion model has to be inverted. The diffusion generator used in the steerable deconvolution is perfectly suited for the regularization of the inversion. It is used quite frequently, for example in [32,33,43,44,84]. Basically, the goal is to find a FOD f such that

$$J(f) = \iint_{\mathbb{R}^3 \times S_2} |(\mathbf{H}f)(\mathbf{x}, \mathbf{n}) - S(\mathbf{x}, \mathbf{n})|^2 \, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{n} + \lambda \iint_{\mathbb{R}^3 \times S_2} (\mathbf{n} \cdot \nabla f)^2 \, \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{n} \tag{57}$$

is minimized, where S denotes the MRI measurement. $\mathbf{H}$ denotes the physical diffusion model, which is usually a spherical convolution, i.e., a convolution on the sphere with a rotationally symmetric kernel. The second term is the regularizer termed in [33] as contour enhancement kernel, or in [84] as fiber continuity (FC). The STA-toolbox provides an easy access to the commonly available kernels and methods. It further provides a positively constrained spherical deconvolution similar to [107]. The following simple example contained in the toolbox creates test data of a 45 fiber crossing configuration and solves the inversion problem:

```
% create the phantom
ds = createTestCrossing(...
      pi/8,pi/4+pi/8,1,0.1)
% compute spherical deconvolution
res = sta_spdeconv(...
    ds.data,ds.dirs,ds.mask,...
    'verbose',true,'lambda',0.01)
```

The result of this example is shown in Fig. 23. Several different spatial and spherical regularization kernels are possible, like the Laplace–Beltrami operator, or the asymmetric contour enhancement (AFC) proposed in [83], which is an extension of the diffusion generator that is used in the example above. It introduces asymmetric orientation features due to intra-voxel curvature of fibers. It can be used via

```
% create bending phantom
ds = createTestBending(5,1,0.01)
% compute asymmetric spherical
% deconvolution
res = sta_spdeconv(....
ds.data,ds.dirs,ds.mask,...
  'verbose',true,'lambda',0.01,...
  'operation','AFC')
```

In Fig. 23, we show qualitative results. Figure 23a, b shows results on a phantom used on the ISBI HARDI reconstruction challenge in 2013. It was created with the Numerical Fiber Generator described in [19]. We show results of a

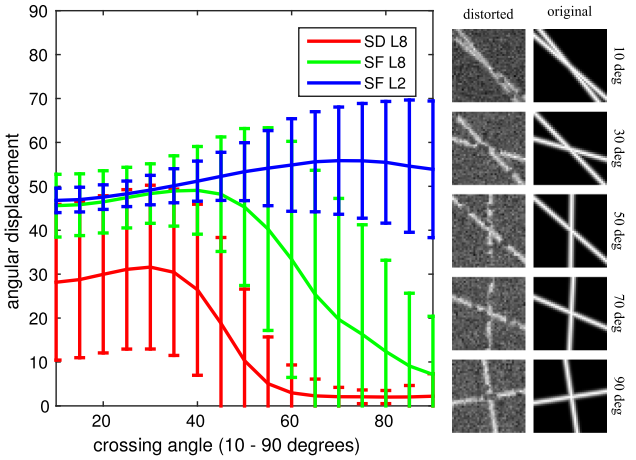

Fig. 24 We randomly created 850 crossings, equally divided into 17 categories. Each category contains two *tubular lines* which are crossing in a certain angle. The angles are ranging from 10° to 90° with a step width of 5°. We corrupted the images with a strong salt-and-pepper-like noise which removed parts from the lines. We additionally added 15% Gaussian noise to the data. We evaluated the orientation functions of three types of tube detection filters at the center. The two largest local maxima of the orientation function were extracted. The error was the mean of the displacement of both maxima in orientation space compared with the "true" line directions. The *plot* shows the mean error over all samples with respect to the crossing degree. We also show the standard deviation as *error bars*. SDL8 is a steerable deconvolution filter of order eight. SFL8 a steerable filter of order 8, and SFL2 a standard steerable filter of order two [3]. As expected, an order two filter cannot resolve crossings

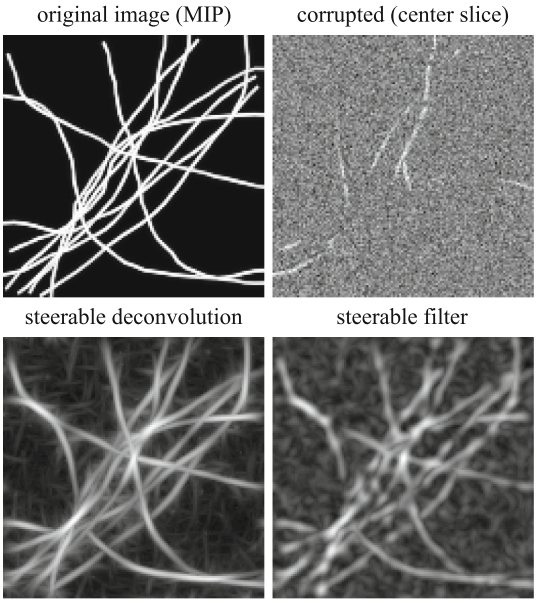

Fig. 25 We show the maximum intensity projection (MIP) of a generated 3D tubular network as it appears in images of neurites or blood vessels. We have corrupted the image with noise. We then applied steerable deconvolution and a steerable filter [3]. The *bottom row* shows the best results (MIP); we could obtain for both approaches

non-regularized spherical deconvolution [107] in comparison with a FC-regularized deconvolution on a low-quality DTI dataset (SNR = 10, #direction = 32). Figure 23d,e shows a more simple crossing example, Fig. 23f,g a bending example using AFC. Finally, in Fig. 23c an in vivo example is shown (61 gradient directions at a b-value of 1 ms/μm^2 at a resolution 2 mm^3 acquired at a Siemens Tim Trio 3T).

6 Conclusion

The main contribution of this article is the provision of an open-source toolbox which implements all the proposed operations involved in spherical tensor algebra. We highlighted the relationship between Cartesian and spherical tensors and introduced the fundamental spherical tensor operations which, in our opinion, are of high value for the image processing community. We complemented the theory with new, implementation relevant insights into the symmetries of spherical bi-spectra. In a survey, we have shown several applications where STA shows implementational advantages over common Cartesian tensor implementations and discretization of the two spheres.

With this toolbox, we want to foster the usage of STA in the context of 3D image processing. The user can avoid the gritty details of the mathematical intricate implementation of spherical tensors operations. The fundamental operations, differentiation and multiplication of spherical tensors are efficiently implemented in C++, OpenMP and CUDA. The API is easily accessible via C++ or MATLAB/OCTAVE. New applications and/or extensions from the community are always welcome!

Acknowledgements This research was supported by the program for Brain Mapping by Integrated Neurotechnologies for Disease Studies (Brain/MINDS) from Japan Agency for Medical Research and development, AMED. This study was supported by Deutsche Forschungsgemeinschaft (German Research Council) via grants DFG RE 3286/2-1 and DFG KI 1089/3-2. We would like to thank Professor Kei Ito from the Department of Computational Biology, The University of Tokyo, for providing us the image of neurite structures in a drosophila fly brain; Fig. 22.

Appendix: Cartesian Tensors

Definition 6 (*Epsilon Tensor*) The epsilon tensor is defined by

$$\epsilon_{ijk} := \begin{cases} 0, & \text{if any two labels are identical} \\ 1, & \text{if i, j, k is an even permutation of 1,2,3} \\ -1, & \text{if i, j, k is an odd permutation of 1,2,3} \end{cases}$$

Spherical two Cartesian Transformation: Let $\mathbf{T}^2$ be a Cartesian tensor of order 2. Then

$$\mathbf{b}^0 = \frac{-\left(t^{(2)}_{00} + t^{(2)}_{11} + t^{(2)}_{22}\right)}{\sqrt{3}}, \mathbf{b}^1 = \begin{pmatrix} \frac{1}{2}\left(t^{(2)}_{20} - t^{(2)}_{02} + \mathbf{i}\left(t^{(2)}_{12} - t^{(2)}_{21}\right)\right) \\ \frac{\mathbf{i}}{\sqrt{2}}\left(t^{(2)}_{01} - t^{(2)}_{10}\right) \\ \frac{1}{2}\left(t^{(2)}_{20} - t^{(2)}_{02} - \mathbf{i}\left(t^{(2)}_{12} - t^{(2)}_{21}\right)\right) \end{pmatrix}$$

$$\mathbf{b}^2 = \begin{pmatrix} \frac{1}{2}\left(t^{(2)}_{00} - t^{(2)}_{11} + \mathbf{i}\left(t^{(2)}_{01} + t^{(2)}_{10}\right)\right) \\ \frac{1}{2}\left((t^{(2)}_{02} + t^{(2)}_{20}) + \mathbf{i}\left(t^{(2)}_{12} + t^{(2)}_{21}\right)\right) \\ \frac{-1}{\sqrt{6}}\left(t^{(2)}_{00} + t^{(2)}_{11} - 2t^{(2)}_{22}\right) \\ \frac{1}{2}\left(-(t^{(2)}_{02} + t^{(2)}_{20}) + \mathbf{i}\left(t^{(2)}_{12} + t^{(2)}_{21}\right)\right) \\ \frac{1}{2}\left(t^{(2)}_{00} - t^{(2)}_{11} - \mathbf{i}\left(t^{(2)}_{01} + t^{(2)}_{10}\right)\right) \end{pmatrix}.$$

are the irreducible representations of $\mathbf{T}^2$ according to (10). The inverse, see (11), is

$$\mathbf{T}^2_{\text{tr}} = -\frac{b^0_0}{\sqrt{3}}\mathbf{I}_{3\times 3},$$

$$\mathbf{T}^2_{\text{anti}} = \begin{pmatrix} 0 & -\mathbf{i}\frac{b^1_0}{\sqrt{2}} & -\frac{1}{2}\left(b^1_{-1} + b^1_1\right) \\ \mathbf{i}\frac{b^1_0}{\sqrt{2}} & 0 & -\mathbf{i}\frac{1}{2}\left(b^1_{-1} - b^1_1\right) \\ \frac{1}{2}\left(b^1_{-1} + b^1_1\right) & \mathbf{i}\frac{1}{2}\left(b^1_{-1} - b^1_1\right) & 0 \end{pmatrix},$$

$$\mathbf{T}^2_{\text{sym}} = \begin{pmatrix} \frac{1}{6}\left(3b^2_{-2} - \sqrt{6}b^2_0 + 3b^2_2\right) & \frac{1}{2}\mathbf{i}\left(b^2_{-2} - b^2_2\right) & \frac{1}{2}\left(b^2_{-1} - b^2_1\right) \\ \frac{1}{2}\mathbf{i}\left(b^2_{-2} - b^2_2\right) & \frac{1}{6}(-3b^2_{-2} - \sqrt{6}b^2_0 - 3b^2_2) & \frac{1}{2}\mathbf{i}(b^2_{-1} + b^2_1) \\ \frac{1}{2}\left(b^2_{-1} - b^2_1\right) & \frac{1}{2}\mathbf{i}\left(b^2_{-1} + b^2_1\right) & \sqrt{\frac{2}{3}}b^2_0 \end{pmatrix}. \tag{58}$$

Irreducible Components

In this paragraph, we give a brief sketch about the composition of Cartesian tensors with a full DOF in terms of their irreducible counterparts. For a proof and further details, we refer to section 4.4 in [15].

The order of all irreducible components of an order j Cartesian is $\leq j$. We start with the most simple tensor with directional information, a spherical tensor of order one. The idea is to, based on order one tensors, recursively construct all possible spherical tensors up to order j. The number and orders of the set of tensors are identical to the number and orders of the irreducible components of an order j Cartesian tensor.

In Sect. 3.1, we have seen that there are, for an order one tensor, three possible operations in each step: decreasing the order (inner product-like operation), keeping the order (cross product-like) and increasing the order (an outer product). With these three operations, we set up Algorithm 1. Table 1 shows the results for tensors up to order 5.

Spherical Harmonics

We always use Racah-normalized spherical harmonics such that $\mathbf{Y}^\ell(\mathbf{r})^\top \mathbf{Y}^\ell(\mathbf{r}) = 1$, or $\mathbf{Y}^\ell(\mathbf{r})^\top \mathbf{Y}^\ell(\mathbf{r}') = P_\ell(\cos(\mathbf{r}, \mathbf{r}'))$,

Algorithm 1 Irreducible Components of Order n

$X = \{\mathbf{b}^1 \in V^1\}$
for $j = 1$ to $(n-1)$ **do**
  $X' = X$
  for each $\mathbf{a}^k$ in X' **do**
    Choose a new $\mathbf{c}^1 \in V^1$
    $X = X \cup \{(\mathbf{a}^k \circ_{k-1} \mathbf{c}^1)\}$
    $X = X \cup \{(\mathbf{a}^k \circ_k \mathbf{c}^1)\}$
    $X = X \cup \{(\mathbf{a}^k \circ_{k+1} \mathbf{c}^1)\}$
  end for
end for
return set of irreducible tensors X

Table 1 Relation between Cartesian and spherical (irreducible) tensors

Spherical	Order: 0	1	2	3	4	5	
	DOF: 1	3	5	7	9	11	
Cartesian	Number of spherical tensors						Total
OrderMax DOF							
0 1	1						1
1 3	0	1					1
2 9	1	1	1				3
3 27	1	3	2	1			7
4 81	3	6	6	3	1		19
5 243	6	15	15	10	4	1	51

There always exists a unitary transformations from Cartesian tensor space to the spherical tensor space. This transformation decomposes a Cartesian tensor into a, quite large, number of spherical counterparts

where the P_ℓ are the Legendre polynomials:

$$P_\ell(t) = \frac{1}{2^\ell \ell!}\partial_t^\ell (t^2-1)^\ell. \tag{59}$$

In terms of the associated Legendre polynomials, the components Y_m^ℓ of the spherical harmonics are written as

$$Y_m^\ell(\phi,\theta) = \sqrt{\frac{(l-m)!}{(l+m)!}} P_\ell^m(\cos(\theta)) e^{\mathbf{i}m\phi}. \tag{60}$$

The Racah-normalized spherical harmonics are orthogonal with respect to

$$\int_{S_2} \overline{Y_{m_1}^{\ell_1}}(\mathbf{n}) Y_{m_2}^{\ell_2}(\mathbf{n}) d\mathbf{n} = \frac{4\pi}{2j+1}\delta_{j_1,j_2}\delta_{m_1,m_2}. \tag{61}$$

They can be turned into the orthonormal spherical harmonics via $\sqrt{\frac{2j+1}{4\pi}} Y_m^\ell$.

Mostly, we write $\mathbf{r} \in S^2$ instead of (ϕ,θ). The Racah-normalized solid harmonics[4] can be written as

$$R_m^\ell(\mathbf{r}) = \sqrt{(\ell+m)!(\ell-m)!} \times \sum_{i,j,k} \frac{\delta_{i+j+k,\ell}\delta_{i-j,m}}{i!\,j!k!2^i 2^j}(x-\mathbf{i}y)^j(-x-\mathbf{i}y)^i z^k, \tag{62}$$

where $\mathbf{r} = (x,y,z)$. They are related to spherical harmonics[5] by $R_m^\ell(\mathbf{r})/r^\ell = Y_m^\ell(\mathbf{r})$.

The spherical harmonics rotate according to

$$\mathbf{Y}^j(\mathbf{U}(g)\mathbf{n}) = \mathbf{D}(g)^j \mathbf{Y}^j(\mathbf{n}). \tag{63}$$

Coupling two spherical harmonics with each other gives another spherical harmonic of desired order:

$$\mathbf{Y}^{j_1}(\mathbf{n}) \circ_j \mathbf{Y}^{j_2}(\mathbf{n}) = \langle j0|j_1 0, j_2 0\rangle \mathbf{Y}^j(\mathbf{n}). \tag{64}$$

$$\langle J0|\ell_1 0, \ell_2 0\rangle Y_M^J = \sum_{m_1,m_2} \langle JM|\ell_1 m_1, \ell_2 m_2\rangle Y_{m_1}^{\ell_1} Y_{m_2}^{\ell_2} \tag{65}$$

$$\langle \ell_2 0|\ell_1 0, J0\rangle Y_M^J = \sum_{m_1,m_2} \langle \ell_2 m_2|\ell_1 m_1, JM\rangle \overline{Y_{m_1}^{\ell_1}} Y_{m_2}^{\ell_2} \tag{66}$$

$$Y_{m_1}^{\ell_1} Y_{m_2}^{\ell_2} = \sum_{J,M} \langle JM|\ell_1 m_1, \ell_2 m_2\rangle \langle J0|\ell_1 0, \ell_2 0\rangle Y_M^J \tag{67}$$

$$\overline{Y_{m_1}^{\ell_1}} Y_{m_2}^{\ell_2} = \sum_{J,M} \frac{2J+1}{2\ell_1+1} \langle \ell_1 m_1|\ell_2 m_2, JM\rangle \langle \ell_1 0|\ell_2 0, J0\rangle Y_M^J \tag{68}$$

Spherical Expansion of the Dirac Delta Function Let $\mathbf{n}, \mathbf{n}' \in S_2$ and $\delta_{\mathbf{n}}^2 : S_2 \to \mathbb{R}$ the delta function on the 2-sphere, whereas $\delta_{\mathbf{n}}^2(\mathbf{n}') = \delta(\theta-\theta')\delta(\phi-\phi')$ and $\int_{S_2} \delta_{\mathbf{n}}^2(\mathbf{n}') d\mathbf{n}' = 1$. According to [7], page 792,

$$\delta_{\mathbf{n}}^2(\mathbf{n}') := \sum_{j=0}^{\infty} \tfrac{(2j+1)}{4\pi} (\mathbf{Y}^j(\mathbf{n}'))^T \mathbf{Y}^j(\mathbf{n}). \tag{69}$$

The Plane Wave The plane wave expansion (see e.g [88], p. 136) in terms of spherical harmonics:

$$e^{\mathbf{i}\mathbf{k}^T\mathbf{r}} = \sum_{j=0}^{\infty} (\mathbf{i})^j (2j+1) J_j(kr) \mathbf{Y}^j(\mathbf{r})^T \mathbf{Y}^j(\mathbf{k}). \tag{70}$$

[4] The Wolfram Functions Site, http://functions.wolfram.com/05.10.06.0027.01.

[5] The Wolfram Functions Site, http://functions.wolfram.com/05.10.23.0008.01.

Clebsch–Gordan Coeffcients

The Clebsch–Gordan coefficients written in terms binomial coefficients[6] are defined by

$$\begin{aligned}\langle j_1 m_1, j_2 m_2 | j m\rangle &= \delta_{m,(m_1+m_2)} \\ &\frac{\sqrt{\binom{2j_1}{-j+j_1+j_2}\binom{2j_2}{-j+j_1+j_2}}}{\sqrt{\binom{j+j_1+j_2+1}{-j+j_1+j_2}\binom{2j_1}{j_1-m_1}\binom{2j_2}{j_2-m_2}\binom{2j}{j-m}}} \\ &\times \sum_{k=\max(0,-j+j_2-m_1,-j+j_1+m_2)}^{k=\min(-j+j_1+j_2,j_1-m_1,j_2+m_2)} (-1)^k \\ &\times \binom{-j+j_1+j_2}{k}\binom{j+j_1-j_2}{-k+j_1-m_1}\binom{j-j_1+j_2}{-k+j_2+m_2}; \end{aligned} \tag{71}$$

see also [1].

The Clebsch–Gordan coefficients of $SO(3)$ fulfill several orthogonality relations:

$$\begin{aligned}&\sum_{j,m}\langle jm|j_1m_1, j_2m_2\rangle\langle jm|j_1m_1', j_2m_2'\rangle \\ &\quad = \delta_{m_1,m_1'}\delta_{m_2,m_2'} \end{aligned} \tag{72}$$

$$\begin{aligned}&\sum_{j,m}\frac{2j+1}{2j_1+1}\langle j_1m_1|jm, j_2m_2\rangle\langle j_1m_1'|jm, j_2m_2'\rangle \\ &\quad = \delta_{m_1,m_1'}\delta_{m_2,m_2'} \end{aligned} \tag{73}$$

$$\begin{aligned}&\sum_{m=m_1+m_2}\langle jm|j_1m_1, j_2m_2\rangle\langle j'm'|j_1m_1, j_2m_2\rangle \\ &\quad = \delta_{j,j'}\delta_{m,m'} \end{aligned} \tag{74}$$

$$\begin{aligned}&\sum_{m_1,m}\langle jm|j_1m_1, j_2m_2\rangle\langle jm|j_1m_1, j_2'm_2'\rangle \\ &\quad = \frac{2j+1}{2j_2'+1}\delta_{j_2,j_2'}\delta_{m_2,m_2'} \end{aligned} \tag{75}$$

For particular combinations, there are simple, explicit formulas:

$$\begin{aligned}&\langle \ell m|(\ell-\lambda)(m-\mu),\lambda\mu\rangle = \\ &\quad \binom{\ell+m}{\lambda+\mu}^{1/2}\binom{\ell-m}{\lambda-\mu}^{1/2}\binom{2\ell}{2\lambda}^{-1/2} \end{aligned} \tag{76}$$

$$\begin{aligned}&\langle \ell m|(\ell+\lambda)(m-\mu),\lambda\mu\rangle = \\ &\quad \times(-1)^{\lambda+\mu}\binom{\ell+\lambda-m+\mu}{\lambda+\mu}^{1/2} \\ &\quad \times\binom{\ell+\lambda+m-\mu}{\lambda-\mu}^{1/2}\binom{2\ell+2\lambda+1}{2\lambda}^{-1/2} \end{aligned} \tag{77}$$

[6] The Wolfram Functions Site, http://functions.wolfram.com/07.38.06.0003.01.

There are several symmetry relations

$$\langle jm|j_1m_1, j_2m_2\rangle = \langle j_1m_1, j_2m_2|jm\rangle \tag{78}$$

$$\langle jm|j_1m_1, j_2m_2\rangle = (-1)^{j+j_1+j_2}\langle jm|j_2m_2, j_1m_1\rangle \tag{79}$$

$$\begin{aligned}&\langle jm|j_1m_1, j_2m_2\rangle \\ &\quad = (-1)^{j+j_1+j_2}\langle j(-m)|j_1(-m_1), j_2(-m_2)\rangle \end{aligned} \tag{80}$$

$$\begin{aligned}&\langle jm|j_1m_1, j_2m_2\rangle \\ &\quad = \sqrt{\frac{2j+1}{2j_2+1}}(-1)^{j_1+m_1}\langle j_2m_2|jm, j_1(-m_1)\rangle, \end{aligned} \tag{81}$$

and associativity relations:

$$\begin{aligned}&\langle J, M|j_1+j_2, m_1+m_2, j_3, m_3\rangle \\ &\quad\times\langle j_1+j_2, m_1+m_2|j_1, m_1, j_2, m_2\rangle \\ &= \langle J, M|j_1+j_3, m_1+m_3, j_2, m_2\rangle \\ &\quad\times\langle j_1+j_3, m_1+m_3|j_1, m_1, j_3, m_3\rangle \end{aligned} \tag{82}$$

where $J = j_1 + j_2 + j_3$ and $M = m_1 + m_2 + m_3$. For $j_3 > j_1 + j_2$, there exist further associativities, namely

$$\begin{aligned}&\langle j_3-j_1-j_2, m_1+m_2+m_3| \\ &\quad J-j_1, m_1+m_3, j_2, m_2\rangle \\ &\quad\times\langle j_3-j_1, m_1+m_3|j_1, m_1, J, m_3\rangle \\ &= \langle j_3-j_1-j_2, m_1+m_2+m_3| \\ &\quad J-j_2, m_2+m_3, j_1, m_1\rangle \\ &\quad\times\langle j_3-j_2, m_2+m_3|j_2, m_2, J, m_3\rangle \end{aligned} \tag{83}$$

Wigner 6j-Symbols

Definition 7 (*Wigner 6j-Symbols*) The Wigner 6j-Symbols are defined by

$$\begin{aligned}&\begin{Bmatrix} j_1 & j_2 & j_4 \\ J & j_3 & j_5 \end{Bmatrix} = \frac{(-1)^{j_1+j_2+j_3+J}}{\sqrt{(2j_4+1)(2j_5+1)}} \\ &\quad\times\sum_{\substack{m_1,m_2,m_3\\ m_4,m_5}}\langle j_1m_1, j_2m_2|j_4m_4\rangle\langle j_3m_3, j_4m_4|JM\rangle \\ &\quad\times\langle j_1m_1, j_3m_3|j_5m_5\rangle\langle j_2m_2, j_5m_5|JM\rangle\,. \end{aligned} \tag{84}$$

(see, e.g., [68], page 1, eq. (D.2))

The permutation of any pairs

$$\begin{Bmatrix} j_1 \\ j_2 \end{Bmatrix} \rightleftarrows \begin{Bmatrix} j_3 \text{or} j_1 \\ j_4 \quad j_2 \end{Bmatrix} \rightleftarrows \begin{Bmatrix} j_2 \\ j_1 \end{Bmatrix} \tag{85}$$

leaves the value of the 6j symbol unaltered (see [68]). Similar to the Clebsch–Gordan coefficients, there exist simple, explicit expressions for some special cases:[7]

$$\begin{Bmatrix} j_1 & j_2 & (j_1+j_2) \\ (j_1+j_2+j_5) & j_5 & (j_1+j_5) \end{Bmatrix} = \frac{(-1)^{2j_1+2j_2+2j_5}}{\sqrt{2(j_1+j_2)+1}\sqrt{2(j_1+j_5)+1}}, \tag{86}$$

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ 0 & j_5 & j_6 \end{Bmatrix} = \frac{(-1)^{j_1+j_2+j_5}\delta_{j_2,j_6}\delta_{j_3,j_5}}{\sqrt{2j_2+1}\sqrt{j_3+1}} \tag{87}$$

and

$$\begin{Bmatrix} j_1 & j_2 & (j_1+j_2) \\ (j_3-j_2-j_1) & j_3 & (j_3-j_1) \end{Bmatrix} = \frac{1}{\sqrt{2(j_1+j_2)+1}\sqrt{2(j_3-j_1)+1}}; \tag{88}$$

a conclusion from Theorem 1.

Wigner-D Matrix

The irreducible representation of $SO(3)$ is called Wigner-D matrices [115,116]. We denote them by the matrix $\mathbf{D}^j(g) \in \mathbb{C}^{(2j+1)\times(2j+1)}$, where $j \in \mathbb{N}_0$, with $j = \{0, \ldots, \infty\}$. The jth order representation works on a $\mathbb{C}^{2j+1}$-dimensional vector space. We denote the components of $\mathbf{D}^j(g)$ by $D^j_{mn}(g)$. In Euler angles in ZYZ convention, we have

$$D^j_{mn}(\gamma,\beta,\alpha) = e^{-\mathbf{i}m\gamma} d^j_{mn}(\beta) e^{\mathbf{i}n\alpha}, \tag{89}$$

where $d^j_{mn}(\beta)$ is the 'small' Wigner-d matrix, which is real-valued and explicitly written as

$$d^j_{mn}(\beta) = [(j+m)!(j-m)!(j+n)!(j-n)!]^{1/2} \sum_s \frac{(-1)^{m-n+s}}{(j+n-s)!s!(m-n+s)!(j-m-s)!} \times \left(\cos\frac{\beta}{2}\right)^{2j+n-m-2s} \left(\sin\frac{\beta}{2}\right)^{m-n+2s}. \tag{90}$$

The representations of different orders are connected via the Clebsch–Gordan coefficients by:

$$D^{j_1}_{m_1n_1} D^{j_2}_{m_2n_2} = \sum_{l,m,n} D^j_{mn} \langle jm | j_1m_1, j_2m_2\rangle \langle jn | j_1n_1, j_2n_2\rangle; \tag{91}$$

see equation 2.3.2 in [15].

[7] The Wolfram Functions Site, http://functions.wolfram.com/07.40.03.0017.01, http://functions.wolfram.com/07.40.03.0006.01.

Another important equality is

$$\int_{SO(3)} \overline{D^{j_1}_{m_1n_1}}(g)\overline{D^{j_2}_{m_2n_2}}(g) D^{j_3}_{m_3n_3}(g)\,dg = \frac{8\pi^2}{2j_3+1}\langle j_3m_3 | j_2m_2, j_1m_1\rangle\langle j_3n_3 | j_2n_2, j_1n_1\rangle \tag{92}$$

Relation to Spherical Harmonics The Wigner-D matrices build an orthogonal basis for functions in $SO(3)$. Let $f(\theta,\phi,\psi) : SO(3) \to \mathbb{C}$ be a function with $f(\theta,\phi,\psi) = f(\theta,\phi)$. Let $A^j_{m,n} \in \mathbb{C}$ be the expansion coefficients of f in terms of the Wigner-D matrices. Then the expansion

$$f(\varphi,\theta,\psi) = \sum_{j=0}^{\infty}\sum_{m,n=-j}^{j} \overline{A^j_{m,n}} D^j_{mn}(\varphi,\theta,\psi) = c(j)\sum_{j=0}^{\infty}\sum_{m=-j}^{j} \overline{a}^j_m Y^j_m(\varphi,\theta) \tag{93}$$

is the spherical harmonic expansion (up to a constant $c(j) \in \mathbb{R}$).

Real and Imaginary Tensor Fields

For all spherical tensors, $\mathbf{v}^j \in \mathbb{C}^{2j+1}$ exits a conjugated counterpart $\left(\mathbf{v}^j\right)^{\ddagger} \in \mathbb{C}^{2j+1}$, with $\left(v^j_m\right)^{\ddagger} := (-1)^m \overline{v^j_{-m}}$. The tensor conjugation induces a unique decomposition of the spherical tensor space $\mathbb{C}^{2j+1}$ into two vector spaces $V_j, \mathbf{i}V_j \subset \mathbb{C}^{2j+1}$. Let $\mathbf{v}^j \in \mathbb{C}^{2j+1}$, then

$$\mathbf{v}^j = \underbrace{\frac{(\mathbf{v}^j + \left(\mathbf{v}^j\right)^{\ddagger})}{2}}_{\in V_j} \oplus \underbrace{\frac{(\mathbf{v}^j - \left(\mathbf{v}^j\right)^{\ddagger})}{2}}_{\in \mathbf{i}V_j}. \tag{94}$$

Despite the fact that these vector spaces are complex valued, we treat them as real-valued vector spaces, because they are closed under weighted superposition for the real numbers; i.e., if $\mathbf{v}^j \in V_j$, then $\forall\alpha \in \mathbb{R} : \alpha\mathbf{v}^j \in V_j$. Same for $\mathbf{i}V_j$. With this assumption, the spherical tensor space $\mathbb{C}^{2j+1}$ is a direct sum of these two subspaces, that is $\mathbb{C}^{2j+1} = V_j \oplus \mathbf{i}V_j$. For the sake of consistency to standard complex numbers, we call the vector space $V_j \subset \mathbb{C}^{2j+1}$ the *real* spherical tensor space and $\mathbf{i}V_j \subset \mathbb{C}^{2j+1}$ the *imaginary* spherical tensor space, i.e., we can always represent an $\mathbf{v}^j \in \mathbf{i}V_j$ in terms of an $\mathbf{i}\mathbf{w}^j$, where $\mathbf{w}^j \in V_j$ (and vice versa).

Corollary 2 (Symmetry) *Let $\mathbf{v}^j \in V_j$ and $\mathbf{w}^j \in \mathbf{i}V_j$. The tensors $\mathbf{v}^j$ and $\mathbf{w}^j$ have the following symmetries*

$$v^j_m = (-1)^m \overline{v^j_{-m}} \text{ and } (\textit{real tensor space}) \tag{95}$$

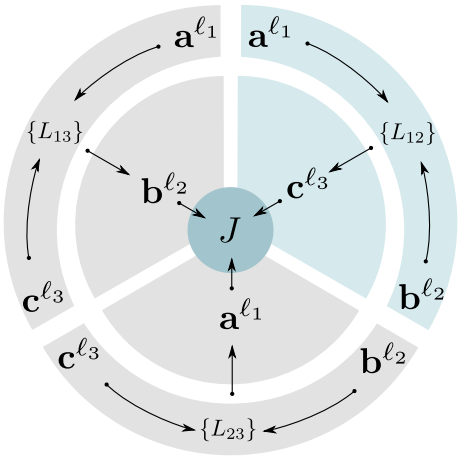

Fig. 26 (Coupling Three Spherical Tensors) The products defined in each third of this circle are spanning the tensor space of the products of the remaining two thirds. Which means they are mutually linear dependent according to Theorem 1

$$w_m^j = (-1)^{m+1}\overline{w_{-m}^j}\ .\ (imaginary\ tensor\ space) \tag{96}$$

This is a direct conclusion from the tensor conjugation property.

Tensor Triple Products

Theorem 1 (Coupling Three Spherical Tensors) *We have the following identity when coupling three spherical tensors* $\mathbf{u}^{j_1} \in \mathbb{C}^{2j_1+1}$, $\mathbf{v}^{j_2} \in \mathbb{C}^{2j_2+1}$ *and* $\mathbf{w}^{j_3} \in \mathbb{C}^{2j_3+1}$ *to form a tensor of rank* J *based on an intermediate rank* $|j_1 - j_2| \leq L_{12} \leq j_1 + j_2$ *:*

$$\begin{aligned}&((\mathbf{u}^{j_1} \circ_{L_{12}} \mathbf{v}^{j_2}) \circ_J \mathbf{w}^{j_3})\\ &= \sum_{L_{23}} (\mathbf{u}^{j_1} \circ_J (\mathbf{v}^{j_2} \circ_{L_{23}} \mathbf{w}^{j_3}))\sqrt{(2L_{12}+1)(2L_{23}+1)}\\ &\times (-1)^{j_1+j_2+j_3+J} \begin{Bmatrix} j_2 & j_1 & L_{12} \\ J & j_3 & L_{23} \end{Bmatrix}.\end{aligned} \tag{97}$$

With $\begin{Bmatrix} j_1 & j_2 & j_4 \\ J & j_3 & j_5 \end{Bmatrix} \in \mathbb{R}$, we denote the Wigner 6j-symbol (see Sect. 1), which are the weighting factors playing a role when coupling three spherical tensors. With Theorem 1, we can identify the symmetries that exist when coupling three spherical tensors. By exchanging the coupling order of the three tensors, we see that each of the following sets of tensors, $\{(\mathbf{u}^{j_1} \circ_J (\mathbf{v}^{j_2} \circ_{L_{23}} \mathbf{w}^{j_3}))\}_{\forall L_{23}}$, $\{(\mathbf{w}^{j_3} \circ_J (\mathbf{u}^{j_1} \circ_{L_{12}} \mathbf{v}^{j_2}))\}_{\forall L_{12}}$ and $\{(\mathbf{v}^{j_2} \circ_J (\mathbf{u}^{j_1} \circ_{L_{13}} \mathbf{w}^{j_3}))\}_{\forall L_{13}}$, can be formed via linear combination of tensors of only one of the remaining sets. That is, they are mutually linearly dependent. This fact is illustrated in Fig. 26. That is, regarding the computation of linearly independent features it is sufficient (and essential) to compute only one set of features out of those three linearly dependent sets. We use this property for computing an linearly independent set of bi-spectrum features in our applications.

Proof In the following, we derive Eq. (97). According to [117], page 17, Eq. (90), there exists the recoupling rule

$$\begin{aligned}&\sum_{M_{12}} \langle j_1 m_1,\ j_2 m_2 | L_{12} M_{12}\rangle \langle L_{12} M_{12},\ j_3 m_3 | J M\rangle\\ &= \sum_{L_{23}, M_{23}} \sqrt{(2L_{12}+1)(2L_{23}+1)}\, W(j_1 j_2 J j_3,\ L_{12} L_{23})\\ &\times \langle j_1 m_1,\ L_{23} M_{23} | J M\rangle \langle j_2 m_2,\ j_3 m_3 | L_{23} M_{23}\rangle,\end{aligned} \tag{98}$$

where $W(j_1 j_2 J j_3, L_{12} L_{23}) \in \mathbb{R}$ is a Racah W-coefficient [78]. Moreover, the following relation to the Wigner 6j-symbols is known (see, e.g., [117], p. 17, Eqs. (93) and (94))

$$\begin{aligned}\begin{Bmatrix} j_3 & L_{12} & J \\ j_1 & L_{23} & j_2 \end{Bmatrix} &= (-1)^{j_1+j_2+j_3+J} W(j_1 j_2 J j_3, L_{12} L_{23})\\ &= \begin{Bmatrix} j_2 & j_1 & L_{12} \\ J & j_3 & L_{23} \end{Bmatrix} \quad \text{(using Eq. 85)}\end{aligned} \tag{99}$$

By just writing out the tensor product of three spherical tensors, and by substituting Eqs. (99) into (98), we can derive the equation in Theorem 1, namely

$$\begin{aligned}&\left[((\mathbf{u}^{j_1} \circ_{L_{12}} \mathbf{v}^{j_2}) \circ_J \mathbf{w}^{j_3})\right]_M = \sum_{M=M_{12}+m_3} u_{m_1}^{j_1} v_{m_2}^{j_2} w_{m_3}^{j_3}\\ &\times \langle L_{12} M_{12},\ j_3 m_3 | J M\rangle \langle j_1 m_1,\ j_2 m_2 | L_{12} M_{12}\rangle\\ &= \sum_{m_3} \sum_{M_{12}} \langle L_{12} M_{12},\ j_3 m_3 | J M\rangle\\ &\times \langle j_1 m_1,\ j_2 m_2 | L_{12} M_{12}\rangle u_{m_1}^{j_1} v_{m_2}^{j_2} w_{m_3}^{j_3}\\ &= \sum_{m_3, L_{23}, M_{23}, L_{23}} \sqrt{(2L_{12}+1)(2L_{23}+1)}(-1)^{j_1+j_2+j_3+J}\\ &\times \begin{Bmatrix} j_2 & j_1 & L_{12} \\ J & j_3 & L_{23} \end{Bmatrix} \langle j_1 m_1,\ L_{23} M_{23} | J M\rangle\\ &\times \langle j_2 m_2,\ j_3 m_3 | L_{23} M_{23}\rangle u_{m_1}^{j_1} v_{m_2}^{j_2} w_{m_3}^{j_3}\\ &= \sum_{L_{23}, m_3, M_{23}} \langle j_1 m_1,\ L_{23} M_{23} | J M\rangle\\ &\times \langle j_2 m_2,\ j_3 m_3 | L_{23} M_{23}\rangle u_{m_1}^{j_1} v_{m_2}^{j_2} w_{m_3}^{j_3}\\ &\times \sqrt{(2L_{12}+1)(2L_{23}+1)}\\ &(-1)^{j_1+j_2+j_3+J} \begin{Bmatrix} j_2 & j_1 & L_{12} \\ J & j_3 & L_{23} \end{Bmatrix}\\ &= \sum_{L_{23}} [(\mathbf{u}^{j_1} \circ_J (\mathbf{v}^{j_2} \circ_{L_{23}} \mathbf{w}^{j_3}))]_M\\ &\times \sqrt{(2L_{12}+1)(2L_{23}+1)}\\ &(-1)^{j_1+j_2+j_3+J} \begin{Bmatrix} j_2 & j_1 & L_{12} \\ J & j_3 & L_{23} \end{Bmatrix}.\end{aligned} \tag{100}$$

□

Proving corollary 1 We show all three equalities using the recoupling rule Eq. (97). In the first scenario, we have

$$\begin{aligned}
&((\mathbf{u}^{j_1} \circ_{(j_1+j_2)} \mathbf{v}^{j_2}) \circ_{(j_1+j_2+j_3)} \mathbf{w}^{j_3}) \\
&\quad = (\mathbf{u}^{j_1} \circ_{(j_1+j_2+j_3)} (\mathbf{v}^{j_2} \circ_{(j_2+j_3)} \mathbf{w}^{j_3})) \\
&\quad \times \underbrace{\sqrt{(2(j_1+j_2)+1)(2(j_2+j_3)+1)} \begin{Bmatrix} j_2 & j_1 & (j_1+j_2) \\ (j_1+j_2+j_3) & j_3 & (j_2+j_3) \end{Bmatrix}}_{=1 \quad \text{(According to Eq. (86))}} \\
&\quad \overset{\text{(using Eq. (78))}}{=} (\mathbf{v}^{j_2} \circ_{(j_1+j_2+j_3)} (\mathbf{u}^{j_1} \circ_{j_1+j_3} \mathbf{w}^{j_3})),
\end{aligned}$$

and in the second scenario

$$\begin{aligned}
&((\mathbf{u}^{j_1} \circ_{(j_1+j_2)} \mathbf{v}^{j_2}) \circ_{(j_3-j_2-j_1)} \mathbf{w}^{j_3}) \\
&\quad = (\mathbf{u}^{j_1} \circ_{(j_3-j_2-j_1)} (\mathbf{v}^{j_2} \circ_{(j_3-j_2)} \mathbf{w}^{j_3})) \\
&\quad \times \underbrace{\sqrt{(2(j_1+j_2)+1)(2(j_3-j_2)+1)} \begin{Bmatrix} j_2 & j_1 & (j_1+j_2) \\ (j_3-j_2-j_1) & j_3 & (j_3-j_2) \end{Bmatrix}}_{=1 \quad \text{(According to Eq. (88))}}.
\end{aligned}$$

Similarly, the third case can be shown:

$$\begin{aligned}
&((\mathbf{u}^{j_1} \circ_{j_3} \mathbf{v}^{j_2}) \circ_0 \mathbf{w}^{j_3}) = (\mathbf{u}^{j_1} \circ_0 (\mathbf{v}^{j_2} \circ_{j_1} \mathbf{w}^{j_3})) \\
&\quad \times \underbrace{(-1)^{j_1+j_2+j_3} \sqrt{(2j_1+1)(2j_3+1)} \begin{Bmatrix} j_2 & j_1 & j_3 \\ 0 & j_3 & j_1 \end{Bmatrix}}_{=1 \quad \text{(According to eq. (87))}}.
\end{aligned}$$

□

References

1. Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Ninth Dover printing, tenth gpo printing edn. Dover, New York (1964)
2. Aganj, I., Lenglet, C., Sapiro, G., Yacoub, E., Ugurbil, K., Harel, N.: Reconstruction of the orientation distribution function in single- and multiple-shell q-ball imaging within constant solid angle. Magn. Reson. Med. **64**, 554–566 (2010)
3. Aguet, F., Jacob, M,. Unser, M.: Three-dimensional feature detection using optimal steerable filters. In: Proceedings of the ICIP, vol. II, pp. 1158–1161, Genova, Italy. IEEE (2005)
4. Allaire, S., Kim, J.J., Breen, S.L., Jaffray, D.A., Pekar, V.: Full orientation invariance and improved feature selectivity of 3D SIFT with application to medical image analysis. In: *Proceedings of the MMBIA*, pp. 1–8 (2008)
5. Antiga, L.: Generalizing vesselness with respect to dimensionality and shape. Insight J. **3** (2007)
6. Applequist, J.: Traceless cartesian tensor forms for spherical harmonic functions: new theorems and applications to electrostatics of dielectric media. J. Phys. A Math. Gen. **22**(20), 4303 (1989)
7. Arfken, G.B., Weber, H.J.: Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide, 6th edn. Academic Press, London (2005)
8. Ballard, Dana H.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognit. **13**(2), 111–122 (1981)
9. Barmpoutis, A., Hwang, M.S., Howland, D., Forder, J.R., Vemuri, B.C.: Regularized positive-definite fourth order tensor field estimation from dw-mri. NeuroImage **45**(1 supplement 1), S153–S162 (2009). (**mathematics in brain imaging**)
10. Barmpoutis, A., Vemuri, B.C., Howland, D., Forder, J.R.: Extracting tractosemas from a displacement probability field for tractography in DW-MRI. In: Metaxas, D., Axel, L., Fichtinger, G., Székely, G. (eds.) Medical Image Computing and Computer-Assisted Intervention—MICCAI 2008, vol. 11, pp. 9–16. Springer (LNCS), New York, NY, USA (2008)
11. Barnett, A.: Theory of q-ball imaging redux: implications for fiber tracking. Magn. Reson. Med. **62**, 910–923 (2009)
12. Bigun, J., Granlund, G.H.: Optimal orientation detection of linear symmetry. In: Proceedings of the ICCV, pp. 433–438, London. IEEE (1987)
13. Bovik, A.C., Clark, M., Geisler, W.S.: Multichannel texture analysis using localized spatial filters. IEEE Trans. Pattern Anal. Mach. Intell. **12**(1), 55–73 (1990)
14. Breiman, Leo: Random forests. Mach. Learn. **45**, 5–32 (2001)
15. Brink, D .M., Satchler, G .R.: Angular Momentum. Oxford Science Publications, Oxford (1993)
16. Buhmann, J., Lange, J., von der Malsburg, C.: Distortion invariant object recognition by matching hierarchically labeled graphs. In: Proceedings of the IJCNN, the International Joint Conference on Neural Networks, pp. 155–159. IEEE (1989)
17. Burgeth, B., Didas, S., Weickert, J.: A General Structure Tensor Concept and Coherence-Enhancing Diffusion Filtering for Matrix Fields, pp. 305–323. Springer, Berlin (2009)
18. Canales-Rodriguez, E.J., Melie-Garcia, L., Iturria-Medina, Y.: Mathematical description of q-space in spherical coordinates: exact Q-ball imaging. Magn. Reson. Med. **61**, 1350–1367 (2009)
19. Caruyer, E., Daducci, A., Descoteaux, M., Houde, J., Thiran, J., Verma, R.: Phantomas: a flexible software library to simulate diffusion MR phantoms. In: ISMRM. Milan, Italy. https://hal.inria.fr/hal-00944644 (2014)
20. Cheung, W., Hamarneh, G.: N-SIFT: n-dimensional scale invariant feature transform for matching medical images. In: Proceedings of the ISBI, pp. 720–723, Washington, DC. IEEE (2007)
21. Cheung, W., Hamarneh, G.: Scale invariant feature transform for n-dimensional images (n-SIFT). Insight J. (2007)
22. Chirikjian, Gregory S., Kyatkin, Alexander B.: An operational calculus for the Euclidean motion group with applications in robotics and polymer science. J. Fourier Anal. Appl. **6**(6), 583–606 (2000)
23. Chirikjian, G.S., Wang, Y.: Engineering Applications of the Motion-Group Fourier Transform. MSRI Publications, Berkeley, CA, USA (2003)
24. Choorat, P., Chiracharit, W., Chamnongthai, K.: A single tooth segmentation using structural orientations and statistical textures. In: Proceedings of the Biomedical Engineering International Conference (BMEiCON), 2011, pp. 294–297. IEEE (2012)
25. Di Claudio, E.D., Jacovitti, G., Laurenti, A.A.: Maximum likelihood orientation estimation of 1-D patterns in Laguerre-Gauss subspaces. IEEE Trans. Image Process. **19**, 1113–1125 (2010)
26. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the CVPR, pp. 886–893. IEEE (2005)
27. Daugman, J.G.: Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. J. Opt. Soc. Am. A Opt Image Sci Vis. **2**(7), 1160–1169 (1985)
28. Delputte, S., Dierckx, H., Fieremans, E., D'Asseler, Y., Achten, E., Lemahieu, I.: Postprocessing of brain white matter fiber orientation distribution functions. In: ISBI'07, pp. 784–787 (2007)
29. Duits, R., Franken, E.: Left-invariant diffusions on the space of positions and orientations and their application to crossing-preserving smoothing of HARDI images. Int. J. Comput. Vis. **92**, 231–264 (2011)
30. Duits, R., Franken, E.M.: Line enhancement and completion via left-invariant scale spaces on SE(2). In: Lecture Notesof Computer Science, Proceedings 2nd International Conference on Scale Space and Variational Methods in Computer Vision, vol. 5567, pp. 795–807 (2009)

31. Duits, R. Franken, E.M.:Left invariant parabolic evolution equations on SE(2) and contour enhancement via invertible orientation scores, part i: Linear left-invariant diffusion equations on SE(2), part ii: nonlinear left-invariant diffusion equations on invertible orientation scores. Q. Appl. Math. AMS. **68**(2), 255–292 (2010)
32. Duits, R., Führ, H., Janssen, B., Bruurmijn, M., Florack, L., van Assen, H.: Evolution equations on Gabor transforms and their applications. Appl. Comput. Harmon. Anal. **35**(3), 483–526 (2013)
33. Duits, R., Haije T.C.J.D., Ghosh, A., Creusen, E., Vilanova, A., ter Haar Romeny, B.: Fiber enhancement in diffusion-weighted MRI. In: Scale Space and Variational Methods in Computer Vision, pp 1–13. Springer (2011)
34. Fehr, J.: Local rotation invariant patch descriptors for 3D vector fields. In: Proceedings of the ICPR, pp. 1381–1384. IEEE, Istanbul, Turkey (2010)
35. Fehr, J., Burkhardt, H.: Phase based 3D texture features. In: Proceedings of the DAGM, pp. 263–272, Berlin, Germany. LNCS, Springer (2006)
36. Fehr, J., Burkhardt, H.: Harmonic shape histograms for 3D shape classification and retrieval. In: IAPR Workshop on Machine Vision Applications (MVA2007). IEEE Computer Society (2007)
37. Fehr, J., Reisert, M., Burkhardt, H.: Cross-correlation and rotation estimation of local 3D vector field patches. In: Proceedings of the 5th International Symposium on Advances in Visual Computing: part I, ISVC '09, pp. 287–296. Springer (2009)
38. Flitton, G., Breckon, T.P., Megherbi, N., Cranfield, UK.: Object recognition using 3D SIFT in complex CT volumes. In: Proceedings of the BMVC, pp. 11–1, Aberystwyth, UK, BMVA (2010)
39. Florack, L.M.J., ter Haar Romeny, B.M., Koenderink, J.J., Viergever, M.A.: General intensity transformations and second order invariants. In: Theory and Applications of Image Analysis: Selected Papers from the 7th Scandinavian Conference on Image Analysis, vol. 2, p. 22. World Scientific Publishing Company Incorporated (1992)
40. Florack, L.M.J., ter Haar Romeny, B.M., Koenderink, J.J., Viergever, M.A.: General intensity transformations and differential invariants. J. Math. Imaging Vis. **4**(2), 171–187 (1994)
41. Förstner, W.: A feature based correspondence algorithm for image matching. Int. Arch. Photogramm. Remote Sens. **26**(3), 150–166 (1986)
42. Frangi, A.F., Niessen, W.J., Vincken, K.L., Viergever, M. A.: Multiscale vessel enhancement filtering. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 130–137. Springer (1998)
43. Franken, Erik, Duits, Remco: Crossing-preserving coherence-enhancing diffusion on invertible orientation scores. Int. J. Comput. Vis. **85**(3), 253–278 (2009)
44. Franken, E, Duits, R.: Scale spaces on the 3D euclidean motion group for enhancement of HARDI data. In: Scale Space and Variational Methods in Computer Vision, pp. 820–831. Springer (2009)
45. Franken, E., van Almsick, M., Rongen, P., Florack, L., ter Haar Romeny, B.:. An efficient method for tensor voting using steerable filters. In: Proceedings of the ECCV 2006, pp. 228–240. Lecture Notes in Computer Science, Springer (2006)
46. Fuster, A., van de Sande, J., Astola, L.J., Poupon, C., Velterop, J., ter Haar Romeny, B.M.: Fourth-order tensor invariants in high angular resolution diffusion imaging. In: Proceedings of the Workshop on Computational Diffusion MRI (CDMRI'11), part of the MICCAI 2011, Toronto, Canada (2011)
47. Gabor, D.: Theory of communication. J. Inst. Elect. Eng. **93**, 429–441 (1846)
48. Ghosh, A., Papadopoulo, T., Deriche, R.: Biomarkers for HARDI: 2nd & 4th order tensor invariants. In: 2012 9th IEEE International Symposium on Biomedical Imaging (ISBI), pp. 26–29. IEEE (2012)
49. Goh, A., Lenglet, C., Thompson, P.M., Vidal, R.: Estimating orientation distribution functions with probability density constraints and spatial regularity. In: Medical Image Computing and Computer-Assisted Intervention—MICCAI 2009, pp. 877–885. Lecture Notes in Computer Science. Springer, Berlin (2009)
50. González, G., Aguet, F., Fleuret, F., Unser, M., Fua, P.: Steerable features for statistical 3D dendrite detection. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 625–632. Springer (2009)
51. Gur, Y., Johnson, C. R.: Generalized HARDI invariants by method of tensor contraction. In: 2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI), pp. 718–721. IEEE (2014)
52. Harris, C., Stephens, M.: A combined corner and edge detector. In: Alvey Vision Conference, vol. 15, p. 50. Manchester, UK (1988)
53. Hoang, T.V.: Représentations d'images pour la reconnaissance de formes. Ph.D. thesis, Université Nancy II (2011)
54. Hoang, T.V., Tabbone, S.: Generic polar harmonic transforms for invariant image description. In: Proceedings of the ICIP, pp. 829–832, Brussels, Belgium, IEEE (2011)
55. Hsu, Y.N., Arsenault, H.H., April, G.: Rotation-invariant digital pattern recognition using circular harmonic expansion. Appl. Opt. **21**(22), 4012–4015 (1982)
56. Hu, P., Zhao, Y., Yang, Z., Wang, J.: Recognition of gray character using Gabor filters. In: Proceedings of the Fifth International Conference on Information Fusion, 2002, vol. 1, pp. 419–424. IEEE (2002)
57. Jain, A.K., Farrokhnia, F.: Unsupervised texture segmentation using Gabor filters. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 14–19 (1990)
58. Jones, Derek K (ed.): Diffusion MRI: Theory, Methods and Applications. Oxford University Press, Oxford (2010)
59. Kainz, B., Keraudren, K., Kyriakopoulou, V., Rutherford, M., Hajnal, J.V, Rueckert, D.: Fast fully automatic brain detection in fetal MRI using dense rotation invariant image descriptors. In: Proceedings of the ISBI, pp. 1230–1233. IEEE (2014)
60. Kainz, B., Malamateniou, C., Murgasova, M., Keraudren, K., Rutherford, M., Hajnal, J.V, Rueckert, D.: Motion corrected 3D reconstruction of the fetal thorax from prenatal MRI. In: Proceedings of the MICCAI, pp. 284–291. Springer (2014)
61. Kakarala, R.: The bispectrum as a source of phase-sensitive invariants for Fourier descriptors: a group-theoretic approach. J. Math. Imaging Vis. **44**(3), 341–353 (2012)
62. Kakarala, Ramakrishna, Mao, Dansheng.: A theory of phase-sensitive rotation invariance with spherical harmonic and moment-based representations. In: Proceedings of the CVPR, pp. 105–112. IEEE (2010)
63. Kamarainen, J.K., Kyrki, V., Kalviainen, H.: Noise tolerant object recognition using Gabor filtering. In: Proceedings of the DSP 2002, the 14th International Conference on Digital Signal Processing, vol. 2, pp. 1349–1352. IEEE (2002)
64. Khotanzad, A., Hong, Y.H.: Invariant image recognition by Zernike moments. IEEE Trans. Pattern Anal. Mach. Intell. **12**(5), 489–497 (1990)
65. Knopp, J., Prasad, M., Willems, G., Timofte, R., Van Gool, L.: Hough transform and 3D SURF for robust three dimensional classification. In: Proceedings of the ECCV, Crete, Greece, Springer (2010)
66. Koenderink, J.J., van Doorn, A.J.: Representation of local geometry in the visual system. Biol. Cybernet. **55**, 367–375 (1987)
67. Koenderink, J.J., van Doorn, A.J.: Generic neighborhood operators. IEEE Trans. Pattern Anal. Mach. Intell. **14**(6), 597–605 (1992)
68. Krainov, V.P., Reiss, H.R., Smirnov, B.M.: Appendix D: Wigner 6j Symbols, in Radiative Processes in Atomic Physics. Radiative Processes in Atomic Physics. John Wiley & Sons, Inc., New York, NY (2005)

69. Kyrki, V., Kamarainen, J.K., Kälviäinen, H.: Simple Gabor feature space for invariant object recognition. Pattern Recogn. Lett. **25**(3), 311–318 (2004)
70. Liu, K., Skibbe, H., Schmidt, T., Blein, T., Palme, K., Brox, T., Ronneberger, O.: Rotation-invariant HOG descriptors using Fourier analysis in polar and spherical coordinates. Int. J. Comput. Vis. **106**(3), 342–364 (2014)
71. Liu, K., Wang, Q., Driever, W., Ronneberger, O.: 2D/3D rotation-invariant detection using equivariant filters and kernel weighted mapping. In: Proceedings of the CVPR, IEEE (2012)
72. Mirzaalian, H., Ning, L., Savadjiev, P., Pasternak, O., Bouix, S., Michailovich, O., Grant, G., Marx, C.E., Morey, R.A., Flashman, L.A., et al.: Inter-site and inter-scanner diffusion MRI data harmonization. NeuroImage **135**, 311–323 (2016)
73. Niemeijer, M., Garvin, M.K., K. Lee, B. van Ginneken, Abràmoff, M.D., Sonka, M.: Registration of 3D spectral OCT volumes using 3D SIFT feature point matching. In: Proceedings of the SPIE, vol. 7259, p. 72591I (2009)
74. Olowoyeye, A., Tuceryan, M., Fang, S.: Medical volume segmentation using bank of Gabor filters. In: Proceedings of the ACM Symposium on Applied Computing, pp. 826–829. ACM (2009)
75. Osada, K., Furuya, T., Ohbuchi, R.: SHREC08 entry: local volumetric features for 3D model retrieval. In: Proceedings of the Shape Modeling and Applications, SMI, pp. 245–246, New York, USA, IEEE (2008)
76. Paganelli, C., Peroni, M., Pennati, F., Baroni, G., Summers, P., Bellomi, M., Riboldi, M.: Scale invariant feature transform as feature tracking method in 4D imaging: a feasibility study. In: Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, pp. 6543–6546. IEEE (2012)
77. Qian, Z., Metaxas, D., Axel, L.: Extraction and tracking of MRI tagging sheets using a 3D Gabor filter bank. In: Proceedings of the IEEE Eng Medicine Biology Society, vol. 1 (2006)
78. Racah, Giulio: Theory of complex spectra. ii. Phys. Rev. **62**, 438–462 (1942)
79. Reisert, M., Burkhardt, H.: Complex derivative filters. IEEE Trans. Image Process. **17**(12), 2265–2274 (2008)
80. Reisert, M., Burkhardt, H.: Equivariant holomorphic filters for contour denoising and rapid object detection. IEEE Trans. Image Process. **17**(2), 190–203 (2008)
81. Reisert, M., Burkhardt, H.: Harmonic filters for generic feature detection in 3D. In: Proceedings of the DAGM, pp. 131–140, Jena, Germany, LNCS. Springer (2009)
82. Reisert, M., Burkhardt, H.: Spherical tensor calculus for local adaptive filtering. In: Aja-Fernández, S., de Luis García, R., Tao, D., Li, X. (eds.) Tensors in Image Processing and Computer Vision, Advances in Pattern Recognition, pp. 153–178. Springer, Berlin (2009)
83. Reisert, M., Kellner, E., Kiselev, V.G.: About the geometry of asymmetric fiber orientation distributions. IEEE Trans. Med. Imaging **31**(6), 1240–1249 (2012)
84. Reisert, M., Kiselev, V.G.: Fiber continuity: an anisotropic prior for ODF estimation. IEEE Trans Med Imaging **30**(6), 1274–1283 (2011)
85. Reisert, M., Mader, I., Anastosoulus, C., Weigel, M., Schnell, S., Kiselev, V.: Global fiber reconstruction becomes practical. Neuroimage **54**(2), 955–962 (2010)
86. Reisert, M., Skibbe, H.: Steerable deconvolution feature detection as an inverse problem. In: Mester, R., Felsberg, M. (eds.) Pattern Recognition, pp. 326–335. Springer (LNCS), Frankfurt/Main, Germany (2011)
87. Reisert, M., Skibbe, H.: Left-invariant diffusion on the motion group in terms of the irreducible representations of SO(3). arXiv:submit/0423757 [math.AP] (2012)
88. Rose, M.: Elementary Theory of Angular Momentum. Dover Publications, Mineola (1995)
89. Sandler, R., Lindenbaum, M.: Optimizing Gabor filter design for texture edge detection and classification. Int. J. Comput. Vis. **84**(3), 308–324 (2009)
90. Savadjiev, P., Campbell, J.S.W., Pike, G.B., Siddiqi, K.: 3D curve inference for diffusion mri regularization and fibre tractography. Med. Image Anal. **10**, 799–813 (2006)
91. Schlachter, M., Reisert, M., Herz, C., Schluermann, F., Lassmann, S., Werner, M., Burkhardt, H., Ronneberger, O.: Harmonic filters for 3D multi-channel data: rotation invariant detection of mitoses in colorectal cancer. IEEE Trans. Med. Imaging **29**(8), 1485–1495 (2010)
92. Schmid, Cordelia, Mohr, Roger: Local grayvalue invariants for image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **19**, 530–535 (1997)
93. Schmidt, T., Pasternak, T., Liu, K., Blein, T., Aubry-Hivet, D., Dovzhenko, A., Duerr, J., Teale, W., Ditengou, F.A., Burkhardt, H., et al.: The iRoCS Toolbox-3D analysis of the plant root apical meristem at cellular resolution. Plant J. **77**(5), 806–814 (2014)
94. Schnell, S., Saur, D., Kreher, B.W., Hennig, J., Burkhardt, H., Kiselev, V.G.: Fully automated classification of HARDI in vivo data using a support vector machine. Neuroimage **46**, 642–651 (2009)
95. Schwab, E., Çetingül, H. E., Afsari, B., Yassa, M.A., Vidal, R.: Rotation invariant features for HARDI. In: International Conference on Information Processing in Medical Imaging, pp. 705–717. Springer (2013)
96. Sheng, Y., Shen, L.: Orthogonal Fourier–Mellin moments for invariant pattern recognition. JOSA A **11**(6), 1748–1757 (1994)
97. Skibbe, H.: *Spherical Tensor Algebra for Biomedical Image Analysis*. Ph.D. thesis, Albert-Ludwigs-Universität Freiburg (2013)
98. Skibbe, H., Reisert, M.: STA-toolkit. Available online: https://bitbucket.org/skibbe/sta-toolbox/
99. Skibbe, H., Reisert, M.: Dense rotation invariant brain pyramids for automated human brain parcellation. In Proceedings of the Informatik 2011, Workshop on Emerging Technologies for Medical Diagnosis and Therapy, Berlin, Germany (2011)
100. Skibbe, H., Reisert, M.: Detection of unique point landmarks in HARDI images of the human brain. In: Proceedings of the Workshop on Computational Diffusion MRI (CDMRI'12), part of the MICCAI 2012, Nice, France (2012)
101. Skibbe, H., Reisert, M., Burkhardt, H.: SHOG—spherical HOG descriptors for rotation invariant 3D object detection. In: *Proc. of the DAGM*, pp. 142–151, Frankfurt, Germany, LNCS. Springer (2011)
102. Skibbe, H., Reisert, M., Ronneberger, O., Burkhardt, H.: Spherical Bessel filter for 3D object detection. In: Proceedings of the ISBI. IEEE, Chicago, Illinois (2011)
103. Skibbe, H., Reisert, M., Schmidt, T., Brox, T., Ronneberger, O. Burkhardt, H.: Fast rotation invariant 3D feature computation utilizing efficient local neighborhood operators. IEEE Trans. Pattern Anal. Mach. Intell. (2012)
104. Sorgi, L., Cimminiello, N., Neri, A.: Keypoints selection in the Gauss Laguerre transformed domain. In: Proceedings of the BMVC, vol. II, p. 539. BMVA, Edinburgh, UK (2006)
105. Suk, T., Flusser, J.: Tensor method for constructing 3D moment invariants. In: Proceedings of the CAIP, pp. 212–219. Springer, Seville, Spain (2011)
106. Thorne, K.S.: Multipole expansions of gravitational radiation. Rev. Mod. Phys. **52**(2), 299 (1980)
107. Tournier, J.-D., Calamante, F., Connelly, A.: Robust determination of the fibre orientation distribution in diffusion MRI: non-negativity constrained super-resolved spherical deconvolution. NeuroImage **35**(4), 1459–1472 (2007)

108. Tournier, J.D., Calamante, F., Gadian, D.G., Connelly, A.: Robust determination of the fibre orientation distribution in diffusion MRI: non-negativity constrained super-resolved spherical deconvolution. NeuroImage **35**(4), 1459–1472 (2007)
109. Tristan-Vega, Antonio, Westin, C.-F.: Estimation of fiber orientation probability density functions in high angular resolution diffusion imaging. NeuroImage **47**(2), 638–650 (2009)
110. Tschumperle, D., Deriche, R.: DT-MRI images: estimation, regularization and application. In: Proceedings of the NeuroImaging Workshop, Eurocast 2003, Las Palmas de Gran Canaria, pp. 46–47. Springer (2003)
111. Tuch, D.S.: Q-ball imaging. Magn. Reson. Med. **52**(6), 1358–1372 (2004). (**English 0740-3194**)
112. Tuch, D.S., Weisskoff, R.M., Belliveau, J.W., Wedeen, V.J.: High angular resolution diffusion imaging of the human brain. In: Proceedings of the ISMRM, Philadelphia, USA, (1999)
113. van Almsick, M.A.: Context Models of Lines and Contours. Ph.D. thesis, Eindhoven University of Technology, Department of Biomedical Engineering, Eindhoven, The Netherlands (2005)
114. Weickert, Joachim: Coherence-enhancing diffusion filtering. Int. J. Comput. Vis. **31**(2–3), 111–127 (1999)
115. Wigner, E.P.: Gruppentheorie und ihre Anwendungen auf die Quantenmechanik der Atomspektren. Vieweg Verlag, Braunschweig (1931)
116. Wigner, E.P., Griffin, J.J.: Group Theory and Its Application to the Quantum Mechanics of Atomic Spectra. Pure and applied Physics. Academic Press, New York (1959)
117. Wormer, P.: Angular momentum theory. Lecture Notes—University of Nijmegen Toernooiveld, 6525 ED Nijmegen, The Netherlands
118. Yang, Bo, Flusser, Jan, Suk, Tomáš: 3D rotation invariants of Gaussian-Hermite moments. Pattern Recognit. Lett. **54**, 18–26 (2015)
119. Yap, P.T., Jiang, X., Kot, A.C.: Two-dimensional polar harmonic transforms for invariant image representation. IEEE Trans. Pattern Anal. Mach. Intell. **32**(7), 1259–1270 (2010)
120. Yoshimura, H., Etoh, M., Kondo, K., Yokoya, N.: Gray-scale character recognition by Gabor jets projection. In: Proceedings of the ICPR, vol. 2, pp 335–338, Barcelona, Spain, IEEE (2000)

Henrik Skibbe is a postdoctoral researcher working at the integrated systems biology laboratory at the Kyoto University in Japan.

Marco Reisert received the Dr.rer.nat. degree in computer science on covariance in kernel methods from the University of Freiburg, Freiburg, Germany, in 2008. In 2009, he started as a postdoctoral researcher with the Medical Physics Department, University Medical Center Freiburg, Freiburg. His interests include 3-D image processing, machine learning and MR imaging/ physics. His main focus is on medical and biological image processing, DTI imaging and DTI-based tractography.