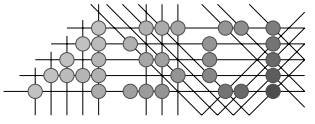
Enhancement of Crossing Elongated Structures in Images

Colophon

The cover has been designed by Erik Franken and Nicole Testerink.

The work described in this thesis was financially supported by the Dutch BSIK program entitled Molecular Imaging of Ischemic heart disease (project number BSIK 03033).

Financial support for the publication of this thesis was kindly provided by the Advanced School for Computing and Imaging (ASCI), and the Technische Universiteit Eindhoven.



This work was carried out in the ASCI graduate school. ASCI dissertation series number 168.

This thesis was typeset by the author using LaTeX2$_\varepsilon$.

Printed by PrintPartners Ipskamp, Enschede, the Netherlands.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-1456-4

Enhancement of Crossing Elongated Structures in Images

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
Rector Magnificus, prof.dr.ir. C.J. van Duijn, voor een
commissie aangewezen door het College voor
Promoties in het openbaar te verdedigen
op maandag 1 december 2008 om 16.00 uur

door

Erik Michiel Franken

geboren te Roosendaal en Nispen

Dit proefschrift is goedgekeurd door de promotor:

prof.dr.ir. B.M. ter Haar Romeny

Copromotor:
dr.ir. R. Duits

Contents

I have only made this letter rather long because I have not had time to make it shorter.

Blaise Pascal (1623–1662)

1 Introduction

1.1 Background

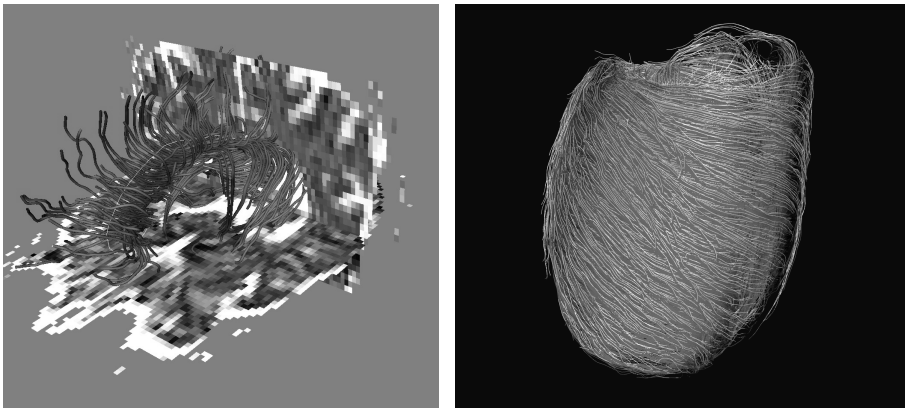
Many fibrous and line-like structures occur in the human body, such as white matter in the brain, muscle and collagen fibres, and blood vessels. Analysis of the properties of these structures is often relevant to make a diagnosis or to answer biomedical research questions. For this purpose, many different biomedical imaging techniques exist to acquire images of these structures.

One of these techniques is *diffusion tensor imaging* (DTI) [102] [9] [10]. DTI is a relatively modern magnetic resonance imaging (MRI) technique for measuring apparent water diffusion processes in fibrous tissue. It is based on the assumption of Brownian motion of water molecules, which is expected to be larger in the direction of fibres. In this way one can for instance measure and visualize white matter tracts in the brain [142], as seen in Figure 1.1(a), which is useful e.g. for preparing brain tumor removal operations. If one knows the positions of the neural fibers, these operations can be performed with minimal damage of neural connections in the brain. The DTI technique is also useful for visualization of muscle structure, especially to visualize the myocardial structure [110], see Figure 1.1(b). Deviations from the expected helical pattern of the myocardium are indications for e.g. heart infarction areas.

Another technique is *microscopy*, which allows to visualize various fibrous structures at a totally different level of scale. This imaging modality is especially relevant in biomedical research. For example, one can acquire microscopic images of sarcomeres [124], which are the basic structures of myofibrils that make up the muscle fibres, see Figure 1.2(a). The overlapping area of two fibres can be seen as a darker band in the image. The regularity of the spacing is a marker for local inhomogeneities in contraction, which can be caused by metabolic deviations such as diabetes. Figures 1.2(b), 1.2(c), and 1.2(d) show other examples of fibrous structures that can be acquired using microscopy techniques.

A third technique is *X-ray fluoroscopy*, a real-time X-ray technique that is used during catheterization procedures. The noisy X-ray fluoroscopy images are necessary to navigate catheters through the body. Figure 1.2(e) shows an example with electrophysiology catheters. These catheters are used to treat patients with heart rhythm disorders. The catheters contain special electrode tips that can be used to acquire internal electrocardiograms and to ablate the spot that causes the cardiac arrhythmias [84]. To assist the medical specialist it is relevant to know the exact positions of the catheters as well as the positions of the electrode tips.

Another example with more line-like rather than fibrous structures are photographs of vessels in the retina, see Figure 1.2(f). It is of interest to segment the vessels, since such segmentation can aid in the screening of diabetic retinopathy, which is a common cause of visual impairment.



(a) Brain (DTI) (b) Heart (DTI)

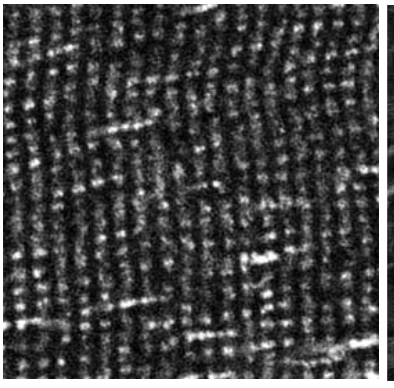
Figure 1.1: Example DTI images containing elongated structures. The visualized fibers are obtained using a fibre tracking algorithm on the DTI data. The left image (a) shows white matter tracts in the brain. The right image (b) shows the myocardial fiber structure. Both images have been generating using the DTI tool [32]. Image (a) taken from [11], image (b) provided by Tim Peeters (Technische Universiteit Eindhoven).

1.2 Project Goal

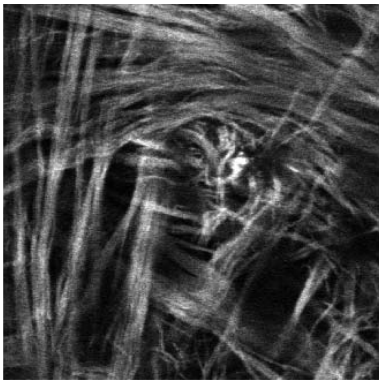
The large number of (bio)medical questions related to images with fibrous and line-like structures suggests the need for automatic processing and analysis of images containing such *elongated structures*. Specifically, it is important to be able to *enhance* these elongated structures and to *detect* them. The acquired (bio)medical images are generally noisy and sometimes the elongated structures are hardly visible. Therefore, enhancement aims at processing the image in order to increase the visibility of the elongated structures. Detection of elongated structures implies that one tries to find the locations of the elongated structures, resulting for instance in a list of coordinates describing the positions of the elongated structures.

In this thesis we will develop a generic framework for processing and analyzing images containing elongated structures. We will focus on generic methods for the *enhancement*. We will only briefly touch upon the detection of elongated structures. We do, however, consider the enhancement of elongated structures an important preprocessing step for subsequent detection of elongated structures, since detection on an enhanced image is often more robust. This implies that relatively simple methods, such as local maxima detection, might already lead to satisfactory results after an enhancement algorithm is applied to the image.

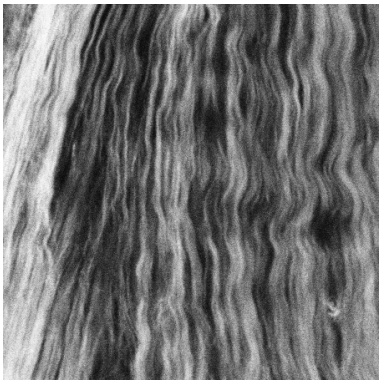
Enhancement and detection of elongated structures is not new. A tremendous



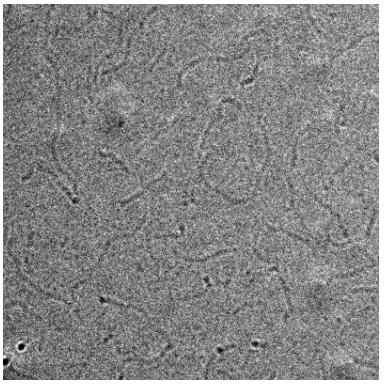
(a) Muscle cells (microscopy)



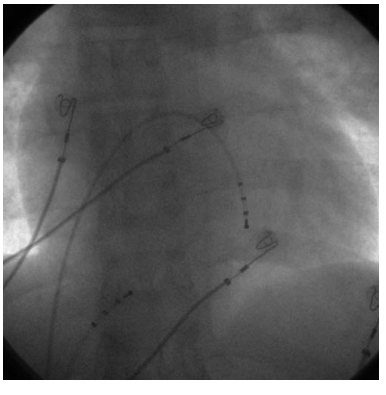
(b) Bone (microscopy)



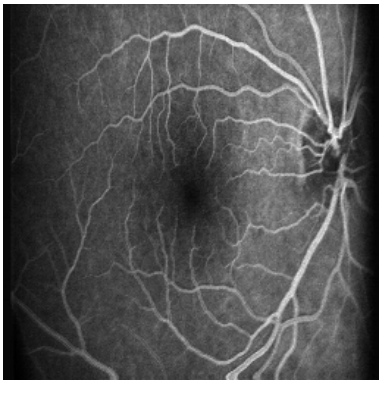
(c) Heart valve (microscopy)



(d) DNA (electron microscopy)



(e) Catheters (X-ray)



(f) Retina (photograph)

Figure 1.2: Examples of (bio)medical images with elongated structures. Images provided by: (a) Jasper Foolen (Technische Universiteit Eindhoven TU/e), (b) Christopher Shaw (University of Birmingham, UK), (c) Mirjam Rubbens (TU/e), (d) Nico Sommerdijk (TU/e), (e) Peter Rongen (Philips Medical Systems), (f) the DRIVE database [126].

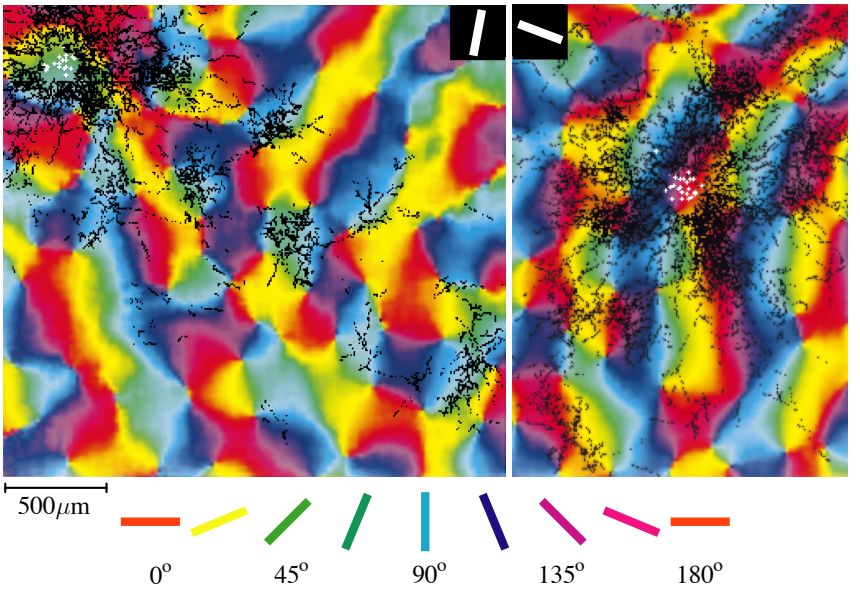
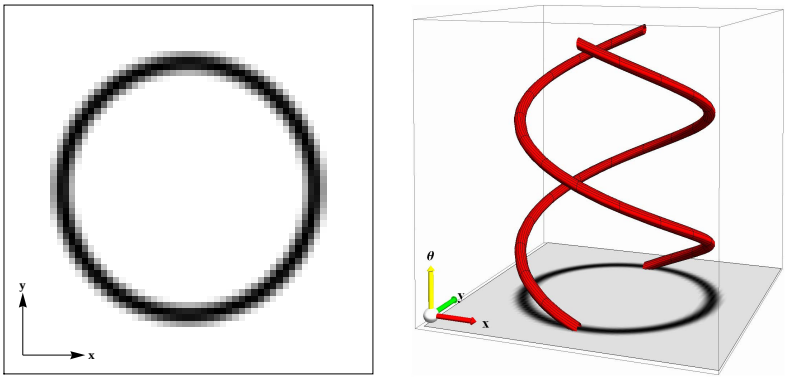


Figure 1.3: Activation patterns in the primary visual cortex of a tree shrew, shown for two different orientation stimuli that were presented to the tree shrew. The orientation sensitivity of the different cells are color-coded. The superimposed white dots (upper left in the left-hand image and in the middle of the right-hand image) indicate cells that took up and transported biocytin, a substance that visualizes connections between cells. The black dots indicate locations of cells that communicated with the cells indicated by the green symbols. Image taken from [14].

amount of methods exist for this purpose, all with their own strengths and weaknesses. However, for most methods it is problematic to handle elongated structures that *cross* each other in the image, since it is usually assumed that at any position in the image there is either no elongated structure or a *single* elongated structure. This results in enhancement artefacts or detection errors at crossings. A detailed investigation of the images in Figure 1.2 reveals that elongated structures cross each other frequently, suggesting that appropriate handling of crossings *is* relevant. Therefore, in this thesis we focus on the development of generically applicable methods for enhancement of elongated structures that can deal with crossings.

1.3 Approach

For almost all vision tasks the performance of the human visual system is still superior compared to contemporary algorithms on computers. Therefore, it makes



(a) Image (b) Orientation score

Figure 1.4: Illustration of orientation score construction of an image containing a single circle. Since the orientation changes linearly as one "travels" over the circle, the resulting response in the three-dimensional orientation score is confined to a spiral. The spirals in this illustration can be seen as iso-surfaces of the orientation score constructed from the circle image. Note that the orientation dimension (displayed vertically) is in fact 2π-periodic.

sense to ask the questions: how do the "algorithms" in the biological visual system work? And can we imitate this by computer algorithms?

The biological visual system has been extensively investigated in various mammalian species, [72, 83]. Experiments revealed that in the primary visual cortex, which is an important area in the brain for processing visual information, many neurons excite a response to elongated structures in the field of view that have a specific *orientation*. Furthermore, it has been found that these orientation selective cells have connections with neighboring cells with approximately the same orientation [14]. This neighborhood covers a fairly large part of the primary visual cortex, as shown in Figure 1.3. The communication between these orientation-selective cells over a large spatial distance is probably an important mechanism for the perception of elongated structures.

The primary visual cortex in fact "transforms" the perceived image, which can be seen as a map of luminance values for each spatial position (x, y), to a three-dimensional representation, i.e. a map of orientation confidences for each position and orientation (x, y, θ). This redundant representation of the image simplifies subsequent analysis in higher visual cortical regions. One of the main advantages might be that crossing structures are being *separated*, since at a crossing their orientations are different.

1.3.1 Orientation Scores

The mentioned observation that the biological visual system contains many cells that are strongly orientation-selective, forms an inspiration to use a similar approach in our image processing algorithms. We will use the term *2D orientation score* to refer to any object that maps 2D positions and orientation angles (x, y, θ) to scalar numbers. An intuition of the relation between an elongated structure in a 2D image and the corresponding elongated structure in an orientation score is given by Figure 1.4 and Figure 1.5. We clearly observe that the oriented structures in the images are separated depending on their local orientation.

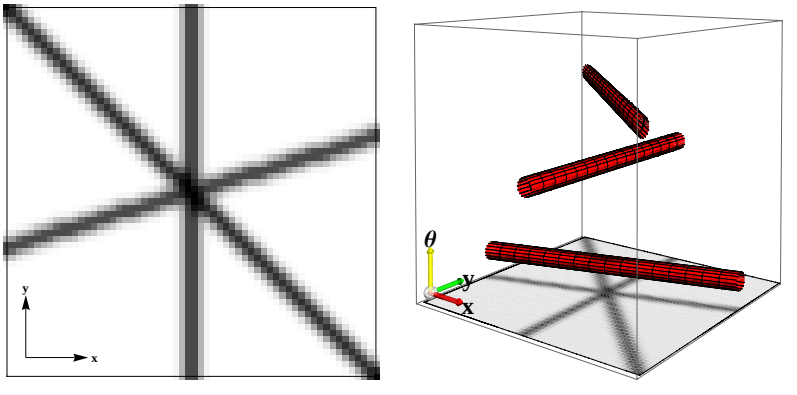
The biological visual system only analyzes two-dimensional images. Depth perception is accomplished by combining the 2D images acquired by the two eyes and is therefore not truly 3D. In many medical images, however, we have three-dimensional volume data. Therefore, we also need to consider *3D orientation scores*, which map 3D positions and orientation angles (x, y, z, β, γ) to scalar numbers. To make it easier to grasp the concepts and to implement the algorithms, we start with 2D orientation scores. Afterwards, we are able to map many concepts to 3D orientation scores.

The concept of orientation scores appears frequently in literature related to image analysis, where often different terms are used for almost the same concepts: e.g. orientation space, orientation channels, and orientation bundle. In this thesis, we will confine ourselves to the term "orientation score". Orientation scores first have occurred in the field of perceptual grouping and biomimicking of the biological visual system [143, 69, 103, 151, 155]. The concept is also applied for *segmenting* crossing structures [21, 7, 137, 117] and estimating local orientation [45].

1.3.2 Invertible Orientation Score Transformations

We will see in this thesis that some imaging modalities directly provide us with an orientation score representation, specifically diffusion tensor imaging, cf. Figure 1.1, and derived methods such as high angular resolution diffusion imaging (HARDI) methods. On the other hand, many other imaging modalities, such as microscopy and X-ray, just render an ordinary 2D or 3D image. In that case, we need an *invertible* transformation to convert such image to an orientation score and vice versa.

Many known image processing algorithms are based on the use of invertible transformations. The idea of such approach is that it simplifies processing of a certain feature of interest in the transformed domain. For example, the Fourier transform makes it easy to manipulate global frequencies in the image, which is applicable for example to determine a certain periodicity in the image. Analogously, the Gabor transform [62] makes it possible to manipulate local frequencies, and wavelet



(a) Image (b) Orientation score

Figure 1.5: Illustration of orientation score construction from an image with crossing lines. Since the lines have different orientations (in this case $\frac{\pi}{10}$, $\frac{\pi}{2}$, and $\frac{3\pi}{4}$), they appear in the orientation score at different positions in the orientation dimension and are therefore torn apart.

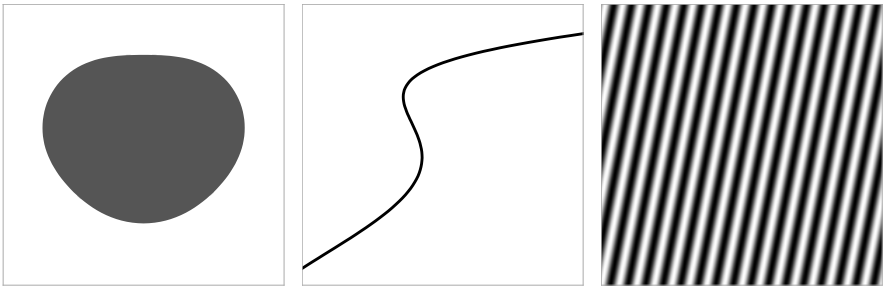
transforms [93] allow to manipulate features at different scales. The idea is exactly the same for invertible orientation score transformations, where *orientation* is the feature of interest. We aim at the construction of operations on the orientation score domain, in order to enhance the *image*, allowing the following chain of operations

$$\text{Transform image to OS} \longrightarrow \text{Process OS} \longrightarrow \text{Transform OS to enhanced image.}$$

Invertible orientation score transformations were first proposed by Kalitzin et al. [81]. Duits et al. [34, 36, 33] developed a theory on the robustness of these transformations. In this thesis, we only briefly touch upon the topic of invertibility, and mainly concentrate on processing operations in the orientation score domain. The invertible orientation scores used in this thesis are based on the aforementioned work by Duits.

1.3.3 Group Theoretical Approach

The orientation score processing methods proposed in this thesis all follow from one important observation: images can be seen as *functions on the translation group*, while orientation scores can be seen as *functions on the Euclidean motion*



(a) contour (curve) (b) line (curve) (c) oriented pattern

Figure 1.6: Illustration of the *elongated structures* occurring in images used throughout this thesis. Both contours (a) and lines (b) are *curves*, which are non-local (i.e, not restricted to a position) 1D graphics primitives. An oriented pattern (c) is a 2D primitive, in which locally the features smoothly change from edge to ridge and vice versa (i.e., the local *phase* varies smoothly).

group[1]. The Euclidean motion group is the group of all rotations and translations and is denoted by $SE(n)$, where n is the dimensionality of the image. Since orientation scores are functions on $SE(n)$ we can use many results from the field of *harmonic analysis on Lie groups* to *map* many well-known image processing concepts to orientation scores. This thesis includes several examples of such mappings, like convolution, linear and nonlinear diffusion, and regularized derivatives. These mappings are not always straightforward, especially since the Euclidean motion group is a noncommutative group, i.e. the order of group transformations are applied matters. Consequently, the implementations are often more complicated.

The idea to explicitly use the properties of the Euclidean motion group for orientation score processing originates from Van Almsick [2] and Duits [34, 33]. Recently, a similar group theoretical approach was adopted by Citti and Sarti [23]. Other Lie groups are applicable as well for image processing in a similar way. For instance, Duits and Janssen [41] use the *Heisenberg* group for processing signals and images via the Gabor domain. Also, the *Galilean* group is used for optic flow estimation [43]. Group theoretical approaches are adopted as well in several other image processing methods [82, 132, 22, 63] and also in other engineering disciplines [22].

This thesis restricts to $SE(2)$ and $SE(3)$ and extends earlier work by Duits and Van Almsick. We aim at developing *algorithms* based on previous results, which were mainly theoretical. The theory by Duits and Van Almsick will be explained in such a way that it should be readable without much prerequisites from the fields of Lie group theory and differential geometry, and, where needed, the theory will be extended.

[1] In Chapter 2 we will define groups and introduce the relevant properties.

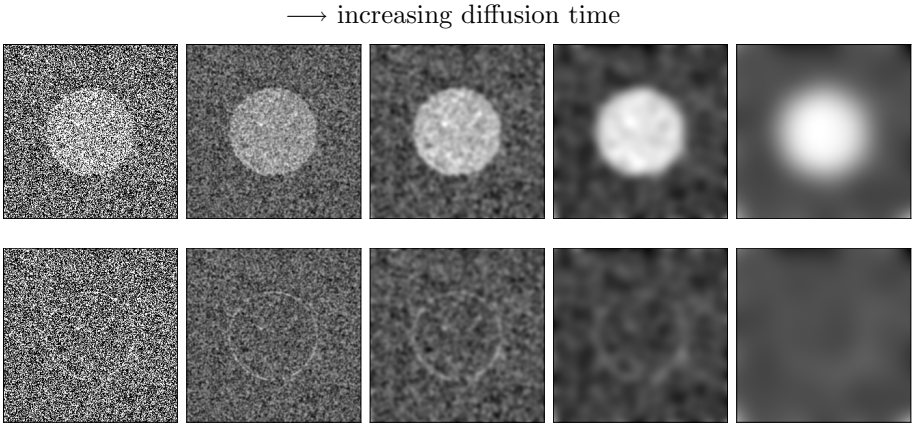
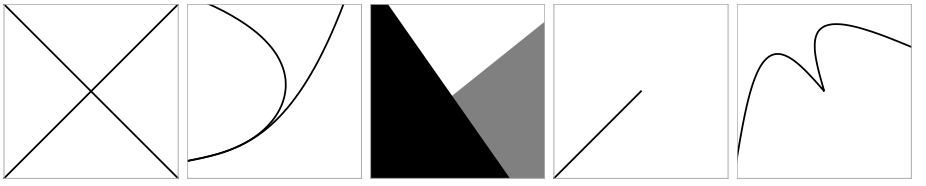


Figure 1.7: Illustration of Gaussian blurring of a contour (upper row) and a line (lower row). On the original image at the left side, the noise makes it difficult to detect the contour resp. line. However, the contour (upper row) remains well-visible as the image is blurred increasingly while the noise is reduced, allowing a robust contour detection at a larger scale. The line (lower row), on the other hand, disappears as more blurring is applied, implying that detection of the line should be performed on a low scale, where noise complicates the task.

1.3.4 Applicability

The applicability of the orientation score framework does not restrict to (bio)medical images of elongated structures with crossings cf. Figures 1.1 and 1.2. The examples we mentioned most often only contain fibrous and line-like structures, which can be roughly classified as either being *lines* or *oriented patterns* cf. Figures 1.6(b) and 1.6(c). We distinguish a third class of oriented structures, namely *contours*, see Figure 1.6(a). For example, contours of organs frequently appear in medical images. Our framework can handle contours in images as well. However, we focus on lines and oriented patterns in this thesis, since these structures are more difficult to analyze using common techniques, as is clarified in Figure 1.7.

Finally, we note that a crossing cf. Figure 1.8(a) is not the only type of singular point that can occur in images with many elongated structures. Figure 1.8(b)-(e) shows other types of singularities that frequently occur. Potentially, our framework can handle these other singular points as well, though they will not be our main concern.



(a) crossing (b) bifurcation (c) T-junction (d) end-point (e) discontinuity in slope

Figure 1.8: Illustrations of the most common singular points that occur in images.

1.4 Structure of this Thesis

The organization of this thesis is illustrated in Figure 1.9. In Chapter 2, orientation scores of 2D images will be introduced in more detail, which provides the necessary theory for the rest of the thesis. We will explain the relevant properties of the Euclidean motion group $SE(2)$, differential geometry in orientation scores, and diffusion equations on orientation scores.

In Chapter 3 the mathematics and the implementation of the $SE(2)$-*convolution* will be described, which is important to operationalize *linear* operations on orientation scores. We will derive how these convolutions can be described in terms of *steerable filters*, which allows for more efficient implementations. Furthermore, we will show how the steerable $SE(2)$-convolution relates to the convolution theorem and the Fourier transform on $SE(2)$.

In Chapter 4 we will use the results of Chapter 3 and show that they apply to tensor image processing. We will explain how *tensor voting* can be put in

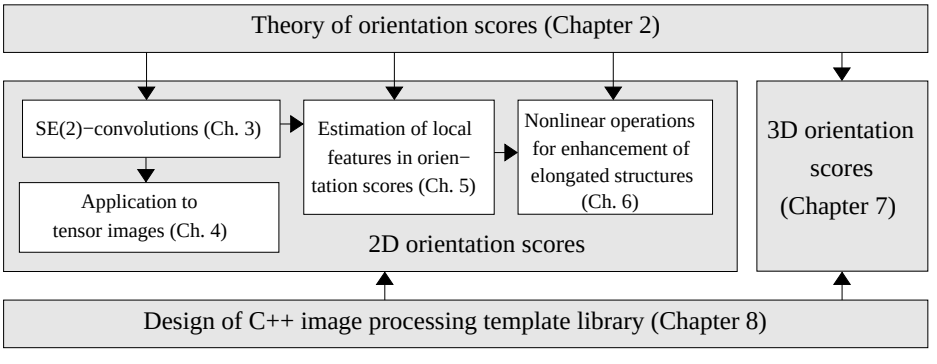


Figure 1.9: Schematic overview of the chapters in this thesis. The arrows indicate "dependencies" between the different chapters.

the orientation score framework and derive *steerable tensor voting*, which uses a simplified steerable $SE(2)$-convolution to perform tensor voting more efficiently. This method will be applied to the detection of electrophysiology catheters in noisy X-ray fluoroscopy images.

Subsequently, Chapter 5 deals with the estimation of local features in orientation scores. These local features describe the local structure at each position in the orientation score. For this purpose, we first discuss how one can take regularized derivatives in orientation scores. This makes it possible to estimate useful local features describing the local anisotropic structure in the orientation score using well-posed derivatives.

Then, in Chapter 6 we describe a method for coherence- and edge-enhancing diffusion filtering of images *via* orientation scores, which is accomplished by a *nonlinear* diffusion equation on the orientation score. The examples will show that this method is able to enhance crossing elongated structures.

Till this point, all theory and implementations are only described for orientation scores of 2D images. In Chapter 7, we describe how all concepts map to 3D. We will show some preliminary results on 3D orientation scores.

Orientation score algorithms have severe computational demands. Although making efficient implementations is not our main aim, the right software design is important as well, to make it possible to do experiments on images with realistic dimensions within a reasonable time span. Therefore, in Chapter 8 we will describe the design of *MathVisionC++*, a library for *any*-dimensional image processing. The applicability of the library is not limited to orientation score processing. We supply implementation and usage examples, and evaluate the speed of some of the library functions.

Finally, in Chapter 9 we will summarize the thesis and discuss possible improvements for future work.

The Theory of Groups is a branch of mathematics in which one does something to something and then compares the result with the result obtained from doing the same thing to something else, or something else to the same thing.

James R. Newman (1907-1966)

2 Orientation Scores

2.1 Introduction

Before we turn to image processing algorithms using orientation scores in later chapters, first we describe the basic theory of orientation scores. We start with describing the important assumption of Euclidean invariance (Section 2.2) and the idea of image processing via orientation scores (Section 2.3). Then, we will introduce the theory on invertible orientation scores (Section 2.4) and explain the design of a practical invertible orientation score, which will be used in the rest of the thesis.

Subsequently, we will treat the basics of group theory and Lie groups (Section 2.5), and we will introduce the 2D Euclidean motion group (Section 2.6). This group theory provides us with essential tools to define sensible operations on orientation scores (Section 2.7). Then, we will explain the basics of differential geometry on orientation scores (Section 2.8).

Finally, we will introduce left-invariant evolution equations on orientation scores (Section 2.9). We will consider the diffusion equation on $SE(2)$, which is a useful equation for curve enhancement, and the convection-diffusion equation, which is a useful equation for curve completion.

This chapter establishes the essential theory needed in the next chapters. The theory is written in such a way that it should be understandable without too many prerequisites from the fields of Lie group theory and differential geometry. For more theoretical underpinning and mathematical details we refer to other publications [39, 40, 36, 35].

In this chapter, and also in large parts of the rest of the thesis, we will assume that images and orientation scores are continuous functions, so we silently ignore the fact that in practical implementations these are always sampled functions. The reason to do so is that it is much more convenient to think in the continuous domain and it is often straightforward to translate it to the discrete domain later on. In the parts of the thesis dealing with implementational details, however, the discrete nature of practical implementations will be made explicit.

2.2 Euclidean Invariant Image Processing Operations

Mathematically, a 2D image f is a function $f : \mathbb{R}^2 \to \mathbb{R}$, which has compact support on the image domain $\Omega = [0, X] \times [0, Y]$, with image dimensions $X, Y \in \mathbb{R}^+$, and which we assume to be square integrable, i.e. $f \in \mathbb{L}_2(\mathbb{R}^2)$. It is straightforward to develop operations on images that are *translation invariant*, i.e. operations that

commute with a translation of the image, meaning that the order in which they are applied does not matter. The formal definition of translation invariance of an operator $\Upsilon : \mathbb{L}_2(\mathbb{R}^2) \to \mathbb{L}_2(\mathbb{R}^2)$ is as follows

$$\forall\, \boldsymbol{x} \in \mathbb{R}^2 :\ \mathcal{T}_{\boldsymbol{x}} \circ \Upsilon = \Upsilon \circ \mathcal{T}_{\boldsymbol{x}}, \tag{2.1}$$

where $\mathcal{T}_{\boldsymbol{x}}$ denotes the translation operation defined by $(\mathcal{T}_{\boldsymbol{x}} f)(\boldsymbol{y}) = f(\boldsymbol{y} - \boldsymbol{x})$ and where "$\circ$" denotes function concatenation, $a \circ b = a(b)$. For example, a convolution of an image f with some filter kernel g,

$$(f *_{\mathbb{R}^2} g)(\boldsymbol{x}) = \int_{\mathbb{R}^2} f(\boldsymbol{x} - \boldsymbol{x}')g(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}', \tag{2.2}$$

always fulfills the property of translation invariance.

One can also create *rotation invariant* operations, meaning that the operation commutes with rotation of the image

$$\begin{aligned} &\forall\, \theta \in \mathbb{R} :\quad \mathcal{R}_\theta \circ \Upsilon = \Upsilon \circ \mathcal{R}_\theta, \\ &\quad \text{with } (\mathcal{R}_\theta f)(\boldsymbol{x}) = f(\mathbf{R}_\theta^{-1}\boldsymbol{x}), \quad \mathbf{R}_\theta = \left(\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right). \end{aligned} \tag{2.3}$$

The latter invariance is important since usually it is not desirable to prefer structures in the image with a certain orientation over structures with other orientations.

Combining translation and rotation invariance we obtain the requirement of *Euclidean invariance* on image processing operation Υ

$$\forall\, \boldsymbol{x} \in \mathbb{R}^2, \forall\, \theta \in \mathbb{R} :\ \mathcal{U}_{(\boldsymbol{x},\theta)} \circ \Upsilon = \Upsilon \circ \mathcal{U}_{(\boldsymbol{x},\theta)}, \quad \text{where } \mathcal{U}_{(\boldsymbol{x},\theta)} = \mathcal{R}_\theta \circ \mathcal{T}_{\boldsymbol{x}}. \tag{2.4}$$

The only *linear* Euclidean invariant operations directly on images are isotropic filtering operations, i.e. convolutions with kernels of the form $K(x, y) = k(\sqrt{x^2 + y^2})$. However, this class of operations is not particularly suitable if the goal is to deal with elongated structures in images. If we consider *nonlinear* operations on images we have more freedom in defining Euclidean invariant operations. An example is the use of gauge coordinates [13] [47], where anisotropic operations are defined by a local coordinate frame that is, ideally, aligned with the most relevant oriented structure. In this thesis we will show that orientation scores provide more possibilities to design Euclidean invariant operations.

2.3 Image Processing via Orientation Scores

If one tries to describe the local characteristics of oriented structures, i.e. lines, edges, and oriented patterns (Figure 1.6) in an image, the coordinates (x, y) in the

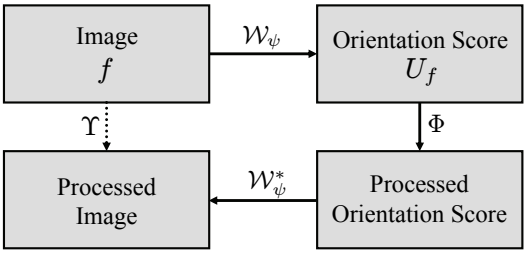


Figure 2.1: A schematic view of image processing via invertible orientation scores. The dotted arrow Υ represents the effective operation on the image. See text for details.

image domain $\mathbb{R}^2$ are not very descriptive, since they only describe the position relative to the horizontal and vertical axes. The codomain is not very descriptive either, since a single grayvalue does not give any useful information on orientation. In an orientation scores, we add a dimension to the image domain, namely orientation. In this way the oriented structures in the image are separated based on their local orientation, as was illustrated in Figure 1.4 and Figure 1.5.

Formally an orientation score U is defined as a function $U : \mathbb{R}^2 \times \mathbb{T} \to \mathbb{R}$ or $\mathbb{C}$, where $\mathbb{R}^2$ corresponds to the image domain and $\mathbb{T}$ is the orientation domain, i.e. $\mathbb{T} = \{e^{i\theta} | \theta \in \mathbb{R}\}$, and where we assume square integrability $U \in \mathbb{L}_2(\mathbb{R}^2 \times \mathbb{T})$. The orientation dimension of a 2D orientation score is periodic, $U(\boldsymbol{x}, \theta) = U(\boldsymbol{x}, \theta + n\,p)$ for all $n \in \mathbb{Z}$, with a periodicity of either $p = 2\pi$ or $p = \pi$. In case of $p = \pi$ one can only represent 180° symmetric oriented structures, which is in many cases sufficient. However, in the explanation of the theory in this thesis we will assume a periodicity of $p = 2\pi$ unless otherwise specified. In the implementations we will use $p = \pi$ whenever possible.

If an orientation score is constructed from an image f by means of a transformation we denote the orientation score as U_f. An orientation score transformation is *invertible* if it is possible to reconstruct image f out of orientation score U_f. In the next section we will explain how to construct invertible orientation scores.

Instead of extending the domain one could think of extending the codomain to describe oriented features, e.g. $\mathbb{R}^2 \to \mathbb{T} \times \mathbb{R}$. The latter approach is substantially different, since in that case each spatial position only maps to a single orientation, while in an orientation score each combination of spatial position and orientation maps to a scalar. The practical advantage of the orientation score approach is manifest: one can transparently handle crossings and bifurcations, as is illustrated in Figure 1.5 on page 8.

The idea is to perform processing operations on 2D images *via* invertible orientation scores as illustrated in Figure 2.1, showing that the net operator Υ on the

image is given by

$$\Upsilon = \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{W}_\psi, \tag{2.5}$$

where $\mathcal{W}_\psi$ denotes the orientation score transformation, $\mathcal{W}_\psi^*$ its inverse, and Φ the operation that is applied on the orientation score. There are many possibilities to develop operations Φ that are sensitive to oriented structures while at the same time the net operation Υ is Euclidean invariant. In Section 2.7 we will derive which class of operators Φ can be used for this purpose.

An important observation that follows from the considerations on Euclidean invariance is that the domains of both images and orientation scores can be considered as *Lie group manifolds*. In fact, $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{U}$ are *representations* of respectively the *translation group*, the *rotation group* and the *Euclidean motion group*. Therefore, Section 2.5 and further sections will give a brief introduction into groups and Lie groups. Subsequently we will explain the properties of the Euclidean motion group. But first, the next section will deal with the construction of invertible orientation scores.

2.4 Invertible Orientation Scores

An invertible orientation score U_f is obtained by correlating the image f with an anisotropic kernel

$$U_f(\boldsymbol{x}, \theta) = (\mathcal{W}_\psi[f])(\boldsymbol{x}, \theta) = (\overline{\mathcal{R}_\theta(\psi)} \star f)(\boldsymbol{x}), \tag{2.6}$$

where $\psi \in \mathbb{L}_2(\mathbb{R}^2)$ is the correlation kernel aligned with the horizontal axis, and the overline denotes complex conjugate. The correlation "$\star$" on $\mathbb{R}^2$ is defined as

$$(h \star f)(\boldsymbol{x}) = \int_{\mathbb{R}^2} h(\boldsymbol{x}' - \boldsymbol{x}) f(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}'. \tag{2.7}$$

The orientation score can also be written using the inner product on $\mathbb{L}_2(\mathbb{R}^2)$,

$$(\mathcal{W}_\psi[f])(\boldsymbol{x}, \theta) = (\mathcal{U}_{(\boldsymbol{x},\theta)}\psi, f)_{\mathbb{L}_2(\mathbb{R}^2)}, \quad \text{with} \quad (h, f)_{\mathbb{L}_2(\mathbb{R}^2)} = \int_{\mathbb{R}^2} \overline{h(\boldsymbol{x})} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{2.8}$$

The exact reconstruction formula accompanying (2.6) is given by

$$\begin{aligned} f(\boldsymbol{x}) &= (\mathcal{W}_\psi^*[U_f])(\boldsymbol{x}) \\ &= \mathcal{F}_{\mathbb{R}^2}^{-1}\left[M_\psi^{-1} \mathcal{F}_{\mathbb{R}^2}\left[\boldsymbol{x} \mapsto \int_0^{2\pi} \int_{\mathbb{R}^2} \psi(\mathbf{R}_\theta^{-1}(\boldsymbol{x} - \boldsymbol{x}')) U_f(\boldsymbol{x}', \theta)(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}' \mathrm{d}\theta \right] \right](\boldsymbol{x}) \\ &= \mathcal{F}_{\mathbb{R}^2}^{-1}\left[M_\psi^{-1} \mathcal{F}_{\mathbb{R}^2}\left[\boldsymbol{x} \mapsto \int_0^{2\pi} \left(\mathcal{R}_\theta(\breve{\psi}) \star U_f(\cdot, \theta) \right)(\boldsymbol{x}) \, \mathrm{d}\theta \right] \right](\boldsymbol{x}), \end{aligned} \tag{2.9}$$

where $\breve{\psi}(\boldsymbol{x}) = \psi(-\boldsymbol{x})$, $\mathcal{F}_{\mathbb{R}^2}$ denotes the Fourier transform on $\mathbb{R}^2$ defined by,

$$\begin{aligned} \mathcal{F}_{\mathbb{R}^2}[f](\boldsymbol{\omega}) &= \frac{1}{2\pi}\int_{\mathbb{R}^2} f(\boldsymbol{x})e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}}\mathrm{d}\boldsymbol{x}, \quad \text{where } \boldsymbol{x}=(x,y), \boldsymbol{\omega}=(\omega_x,\omega_y), \\ \text{inverse transformation } \mathcal{F}^{-1}_{\mathbb{R}^2}[\hat{f}](\boldsymbol{x}) &= \frac{1}{2\pi}\int_{\mathbb{R}^2} f(\boldsymbol{x})e^{i\boldsymbol{\omega}\cdot\boldsymbol{x}}\mathrm{d}\boldsymbol{\omega}. \end{aligned} \tag{2.10}$$

Function $M_\psi : \mathbb{R}^2 \to \mathbb{R}^+$ in (2.9) is given by

$$M_\psi(\boldsymbol{\omega}) = \int_0^{2\pi} \overline{\mathcal{F}_{\mathbb{R}^2}[\mathcal{R}_\theta[\psi]]}\mathcal{F}_{\mathbb{R}^2}[\mathcal{R}_\theta[\psi]]\,\mathrm{d}\theta = \int_0^{2\pi} |\mathcal{F}_{\mathbb{R}^2}[\mathcal{R}_\theta[\psi]]|^2\,\mathrm{d}\theta. \tag{2.11}$$

This function can be seen as a measure for stability of the inverse transformation: the number $M_\psi(\boldsymbol{\omega})$ specifies how well frequency component $\boldsymbol{\omega}$ is preserved by the cascade of construction and reconstruction, if the "compensation term" M_ψ^{-1} would not be included in the reconstruction equation (2.9). It can be verified that the construction / reconstruction equations (2.6) and (2.9) fulfill the following Plancherel's formula

$$\|f\|^2_{\mathbb{L}_2(\mathbb{R}^2)} = \|\mathcal{W}_\psi f\|^2_{M_\psi}, \tag{2.12}$$

where the norm $\|\cdot\|_{M_\psi}$ on the orientation score domain is defined as

$$\|\mathcal{W}_\psi f\|^2_{M_\psi} = \int_{\mathbb{R}^2}\int_0^{2\pi} |(\mathcal{F}_{\mathbb{R}^2}\mathcal{W}_\psi f)(\boldsymbol{\omega},\theta)|^2 \mathrm{d}\theta \frac{1}{M_\psi(\boldsymbol{\omega})}\mathrm{d}\boldsymbol{\omega}, \tag{2.13}$$

where $\mathcal{F}_{\mathbb{R}^2}$ denotes the Fourier transform on the spatial coordinates only. Note that we have $\mathbb{L}_2$-norm preservation, i.e. $\|f\|^2_{\mathbb{L}_2(\mathbb{R}^2)} = \|\mathcal{W}_\psi f\|^2_{M_\psi} = \|\mathcal{W}_\psi f\|^2_{\mathbb{L}_2(SE(2))}$, if and only if $M_\psi = 1$.

Theoretically, reconstruction is well-posed as long as $0 < \delta < M_\psi(\boldsymbol{\omega}) < \infty$ where δ is arbitrarily small. In practice, to prevent numerical problems it is better to aim at $M_\psi(\boldsymbol{\omega}) \approx 1$ for $\|\boldsymbol{\omega}\| < \varrho$, meaning that all frequency components within a ball of radius ϱ in the Fourier domain are preserved. The latter is a natural choice for bandlimited images: because of finite sampling we can assume images to be bandlimited anyway, where the bandwidth coincides with the well-known Nyquist frequency. If $M_\psi \approx 1$ we can approximate the reconstruction by

$$\hat{f}(\boldsymbol{x}) \approx \int_0^{2\pi} \left(\mathcal{R}_\theta[\breve{\psi}] \star U_f(\cdot,\theta)\right)(\boldsymbol{x})\,\mathrm{d}\theta. \tag{2.14}$$

We can even further simplify the reconstruction for filters ψ that satisfy

$$M_\psi(\boldsymbol{\omega}) = \int_0^{2\pi} |\mathcal{F}_{\mathbb{R}^2}[\mathcal{R}_\theta[\psi]](\boldsymbol{\omega})|^2\,\mathrm{d}\theta \approx \int_0^{2\pi} |\mathcal{F}_{\mathbb{R}^2}[\mathcal{R}_\theta[\psi]](\boldsymbol{\omega})|\,\mathrm{d}\theta, \tag{2.15}$$

where the reconstruction formula simplifies to integration over the orientation dimension

$$\hat{f}(\boldsymbol{x}) \approx \int_0^{2\pi} U_f(\boldsymbol{x}, \theta)\, \mathrm{d}\theta, \tag{2.16}$$

which makes this class of filters favorable if computational speed is important.

2.4.1 Design of an Invertible Orientation Score Transformation

The invertibility conditions described in Section 2.4 still allow for a lot of freedom in the selection of a kernel ψ, and it might not be obvious how to choose such a kernel. Therefore, for the duration of this subsection we interrupt the theoretical thread of this chapter to describe a practical design of an orientation score transformation.

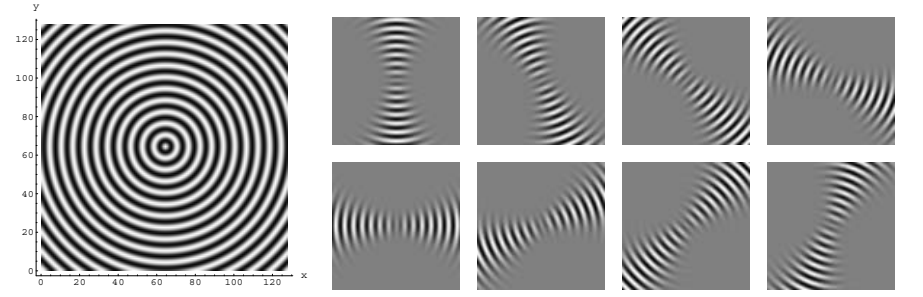
We restrict the possible choices by first formulating a number of practical requirements that our transform should fulfill.

1. The orientation score transform should yield a finite number (N_o) of orientations. This requirement is obvious from an implementational point of view.
2. Reconstruction should be achieved by summing all orientations, i.e.

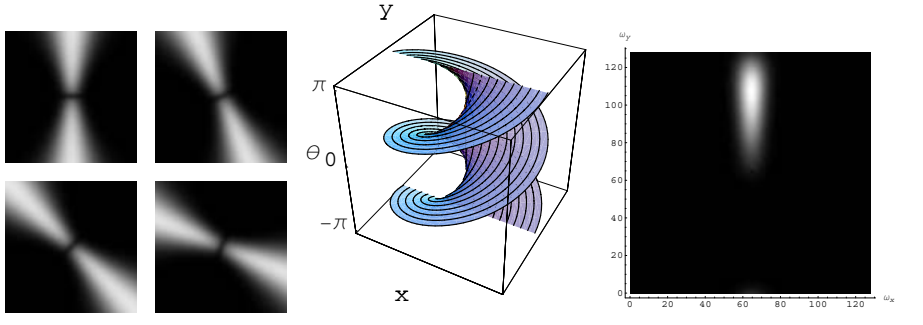
$$f(\boldsymbol{x}) = \sum_{j=0}^{N_o-1} U_f(\boldsymbol{x}, j s_\theta), \tag{2.17}$$

   where s_θ is the orientation sample distance in radians, i.e. $s_\theta = \frac{2\pi}{N_o}$ if the periodicity of the orientation score is 2π.
3. The kernel should be strongly directional, in order to obtain sharp responses on oriented structures. That is, *ideally* an oriented structure at $\boldsymbol{x}$ with orientation θ should only yield a nonzero response at $U_f(\boldsymbol{x}, \theta)$.
4. We want to be able to handle lines, contours, and oriented patterns. This implies that the kernel should pick up edge, ridge, and periodic profiles, see Figure 2.3(a)-(c).
5. In order to pick up local structures, the kernel should be localized in the spatial domain.

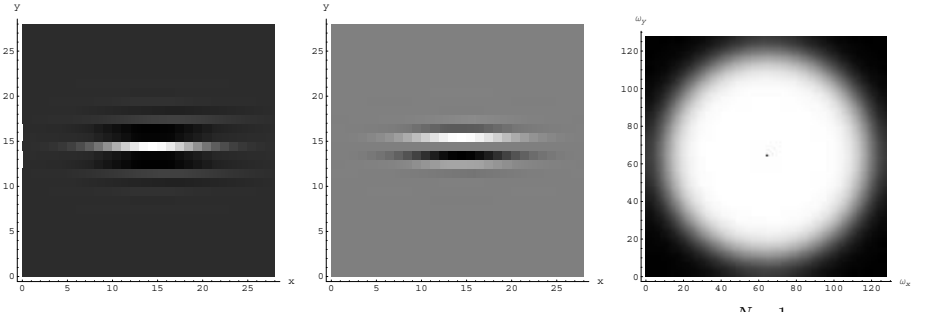
To ensure the kernel to be strongly directional cf. requirement 3, we require the kernel to be a convex cone in the Fourier domain [4]. Furthermore, to be able



(a) Image (b) $\mathrm{Re}(U_f)$



(c) $|U_f|$ (d) Iso-surface of $|U_f|$ (e) $|\mathcal{F}_{\mathbb{R}^2}(\psi)|$



(f) $\mathrm{Re}(\psi)$ (g) $\mathrm{Im}(\psi)$ (h) $\sum_{j=0}^{N_o-1} \mathcal{F}_{\mathbb{R}^2}(\psi)$

Figure 2.2: Example of an orientation score transformation using the kernel of (2.21). (a) Example of an image with concentric circles. (b) Real part of the orientation score U_f displayed for 8 different orientations. (c) The absolute value $|U_f|$ yields a phase-invariant response, displayed for 4 orientations. (d) Sketch of an iso-surface of the absolute value of $|U_f|$ cf. (c), showing that the circles become spirals and all spirals together span a helicoid-shaped plane. (e) Fourier transform of the applied kernel ψ cf. (2.21) for $\theta = 0$. (f) Real part of the kernel with $\theta = 0$. (g) Imaginary part of the kernel. (h) M_ψ (equation (2.11)), which can be seen as the Fourier transform of the net operation if no correction is applied, i.e. if reconstruction equation (2.16) is used. In this example the parameters are set to $k = 2$, $q = 8$, $t = 400$, $\sigma_s = 10$, $N_o = 64$.

to handle both edges and ridges cf. requirement 4, we need to ensure that ψ is a *quadrature filter* [61].

A 1-dimensional filter $k : \mathbb{R} \to \mathbb{C}$ is a quadrature filter if the real and imaginary part of the filter relate to each other by the Hilbert transform $\mathcal{H}$,

$$k(x) = k_{\text{even}}(x) - i\, k_{\text{odd}}(x) = k_{\text{even}}(x) - i\, \mathcal{H}[k_{\text{even}}](x), \tag{2.18}$$

where the Hilbert transform is defined as

$$\mathcal{H}[k_{\text{even}}](x) = \mathcal{F}_{\mathbb{R}}^{-1}[\omega \mapsto i\, \text{sign}(\omega) \mathcal{F}_{\mathbb{R}}[\psi_{\text{even}}](\omega)](x). \tag{2.19}$$

A real-valued signal a filtered with a quadrature filter gives an analytic signal $k*a$, see Figure 2.3, which has the property that one can obtain the instantaneous amplitude by the absolute value $|(k * a)(x)|$, which is phase-invariant, and the instantaneous phase by the argument $\arg((k*a)(x))$. The phase-invariance of the absolute value is useful if we want to treat both edges and ridges in the same way.

Our filter ψ should have the quadrature property in the direction orthogonal to the structures to be detected. In our convention, ψ should pick up structures aligned with the x-axis, so the quadrature property should hold in the y-direction, i.e.

$$\psi(\boldsymbol{x}) = \psi_{\text{even}}(\boldsymbol{x}) - i\, \psi_{\text{odd}}(\boldsymbol{x}), \quad \text{with} \quad \psi_{\text{odd}}(x, y) = \mathcal{H}[\psi_{\text{even}}(x, \cdot)](y). \tag{2.20}$$

Based on the requirements and the considerations above we propose the following kernel

$$\psi(\boldsymbol{x}) = \frac{1}{M} \mathcal{F}_{\mathbb{R}^2}^{-1}[\breve{\psi}](\boldsymbol{x})\, G_{\sigma_s}(\boldsymbol{x}), \tag{2.21}$$

where G_{σ_s} is a Gaussian window that "enforces" spatial locality cf. requirement 5, M is the normalization constant given by $M = N_o \int_{\mathbb{R}^2} \mathcal{F}[\psi](\boldsymbol{\omega}) \mathrm{d}\boldsymbol{\omega}$, which ensures that the reconstructed image has the same grey-value range as the the original image. Function $\breve{\psi}$ is given by

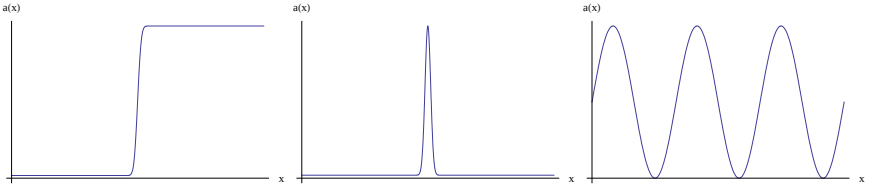
$$\breve{\psi}(\boldsymbol{\omega}) = \breve{\psi}(\rho \cos\varphi, \rho \sin\varphi) = \begin{cases} B_k\left(\frac{(\varphi \mod 2\pi) - \pi/2}{s_\theta}\right) \zeta(\rho) & \rho > 0 \\ \frac{1}{N_o} & \rho = 0, \end{cases} \tag{2.22}$$

where B_k denotes the kth order B-spline given by

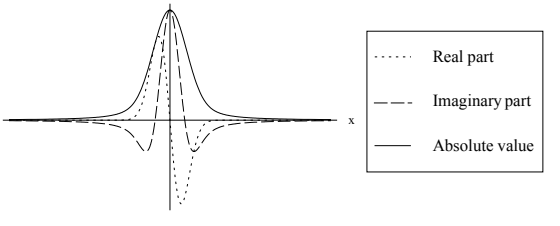
$$B^k(x) = (B^{k-1} * B^0)(x), \qquad B^0(x) = \begin{cases} 1 & \text{if } -1/2 < x < +1/2 \\ 0 & \text{otherwise} \end{cases}, \tag{2.23}$$

and function $\zeta(\rho)$ specifies the radial function in the Fourier domain, chosen as the Gaussian with scale t divided by its Taylor series up to order q to ensure a slower decay, i.e.

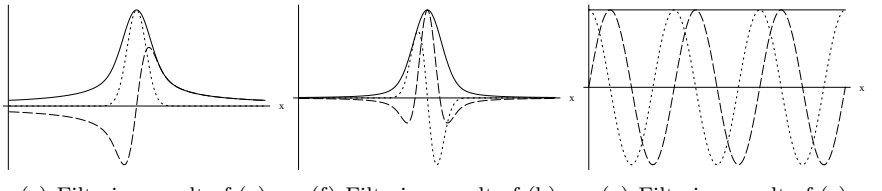
$$\zeta(\rho) = G_t(\rho) \left(\sum_{j=0}^{q} \left(\frac{\mathrm{d}}{\mathrm{d}\rho'} G_t(\rho') \Big|_{\rho'=0} \right) \frac{\rho^j}{j!} \right)^{-1}, \quad G_t(\rho) = \frac{1}{2\sqrt{\pi t}} e^{-\frac{\rho^2}{4t}}. \tag{2.24}$$



(a) Edge (b) Ridge (c) Sine signal a



(d) Quadrature filter k



(e) Filtering result of (a) (f) Filtering result of (b) (g) Filtering result of (c)

Figure 2.3: (a)-(b) Example 1D edge and ridge profile functions, which are the expected profiles orthogonal to contours resp. lines. (c) Sine function, which is an example orthogonal profile to an oriented pattern. (d) Kernel of an example of a 1D quadrature filter. The real and imaginary part relate to each other by the Hilbert transform cf. (2.19). (e)-(g) Results of convolving (a)-(c) with quadrature filter (d), showing that the absolute value is invariant to the phase of the signal. The legend of (d) applies to (e)-(g) as well.

Figure 2.2 shows an example of this orientation score transformation.

Note that some requirements are conflicting to each other, meaning that not all requirements are exactly met. By adjusting the parameters a compromise needs to be found. Particularly, the quadrature property conflicts with the locality requirement, since the spatial multiplicative window G_{σ_s}, which is a convolution in the Fourier domain with $\mathcal{F}_{\mathbb{R}^2}[G_{\sigma_s}]$, causes property (2.20) to only approximately hold dependent on the spatial windows scale σ_s. Note that the quadrature property is also violated because $\check{\psi}(\mathbf{0}) \neq 0$ cf. (2.22). This is needed to reconstruct the average value of the image. If the conflict with the quadrature property lead to problems in practice, they can simply be solved by preprocessing the image to leave out the very low frequencies, i.e. by precalculating

$$\tilde{f}_{\mathrm{DC}} = f *_{\mathbb{R}^2} G_{\tilde{\sigma}_s}, \quad \tilde{f} = f - \tilde{f}_{\mathrm{DC}}, \tag{2.25}$$

where a logical choice is $\tilde{\sigma}_s = \sigma_s$ and where we assume that the calculated $\tilde{f}_{\mathrm{DC}}$ does not contain any relevant information on the elongated structures of interest in the image. We can now construct the orientation score of $\tilde{f}$ instead of f, and after reconstruction we add up $\tilde{f}_{\mathrm{DC}}$ to obtain f again.

The kernel cf. (2.21) is similar to the kernel proposed by Van Ginkel [137, Section 3.4]. The difference is that Van Ginkel does not aim at reconstruction, and therefore he chooses a radial function in the Fourier domain which is optimized to pick up elongated structures and a single scale. We, on the other hand, have to pick up structures at all scales and therefore the radial function ζ is chosen to fill up a large part of the Fourier spectrum, if parameter t is correctly chosen. Furthermore, in angular direction in the Fourier domain van Ginkel uses a Gaussian while we use B-splines, which have the advantage that they are more local and thus improve the directionality.

The invertible orientation score transformation by Kalitzin [80] would have been an alternative option to use. However, this transformation has practical disadvantages, especially the fact that the kernel does not attain its maximum value in the center, i.e. at $\boldsymbol{x} = 0$, leading to orientation scores which represent features in the image in a "squinty way".

2.5 Basics of Group Theory

A key point in our framework is to consider the group structure of the domain of the orientation score. Therefore, in this section we provide the essential basics of (Lie) group theory. More information can be found in several books, e.g. [68, 139].

2.5.1 Definition of a Group

A *group* G is a finite or infinite set of elements with a binary group operation "$\cdot$" (where the infix symbol "$\cdot$" is omitted further on) that fulfills the properties of

- closure: $ab \in G$ for all $a, b \in G$;
- associativity: $(ab)c = a(bc)$ with $a, b, c \in G$;
- identity: there is an $e \in G$ such that $ae = ea = a$ for all $a \in G$;
- inverse: for all $a \in G$ there exists a $a^{-1} \in G$ such that $a^{-1}a = e$;
- Abelian or commutative groups also fulfill the property of commutativity: $ab = ba$ for all $a, b \in G$.

A *subgroup* H is a subset of group elements of a group G that satisfies the group requirements. Note that it therefore always contains the identity element e.

To map the structure of a group to some mathematical object, we need a *representation*. A representation is a mapping of the form $\mathcal{A} : G \to \mathcal{B}(H)$, where H is the linear space to which our mathematical objects belong and $\mathcal{B}(H)$ is the space of bounded linear invertible operators $H \to H$, that maps a group element to an operator, i.e. $\mathcal{A} = (g \mapsto \mathcal{A}_g)$, such that $e \mapsto \mathcal{I}$ (identity element maps to identity operator), $gh \mapsto \mathcal{A}_{gh} = \mathcal{A}_g \circ \mathcal{A}_h$ (group product is preserved), and consequently $g^{-1} \mapsto (\mathcal{A}_g)^{-1}$ (inverse is preserved). Examples of representations will be given in Subsection 2.6.2.

Different types of groups exist. Mainly, we can distinguish between *finite groups*, containing a finite number of elements, and *infinite groups*, containing a infinite number of elements. A *Lie group* is a special type of infinite group, which will be described in the next subsection.

2.5.2 Lie Groups and Lie Algebras

A *Lie group* G is an infinite group where the infinite number of group elements are parameterized by a finite-dimensional differentiable manifold endowed with a metric, i.e. there is a notion of distance between group elements. The fact that the group elements form a manifold makes it possible to use differential calculus on Lie groups.

By looking at the differential properties at the unity element e we obtain the corresponding *Lie algebra* denoted by $T_e(G)$, which is a vector space endowed with a binary operator called the *Lie bracket* or *commutator* $[\cdot, \cdot] : T_e(G) \times T_e(G) \to T_e(G)$ with the following properties

- bilinearity: $[aX + bY, Z] = a[X, Z] + b[Y, Z]$ and $[Z, aX + bY] = a[Z, X] + b[Z, Y]$ for all $X, Y, Z \in T_e(G)$;
- anticommutativity: $[X, Y] = -[Y, X]$ for all $X, Y \in T_e(G)$;
- Jacobi identity: $[X, [Y, Z]] + [Y, [Z, X]] + [Z, [X, Y]] = 0$ for all $X, Y, Z \in T_e(G)$.

The Lie algebra completely captures the local structure of the group. The Lie group and Lie algebra relate to each other by an *exponential mapping* $\exp : T_e(G) \to G$.

Many Lie groups, including all Lie groups used in this thesis, can be expressed as matrix groups. For such a *matrix Lie group*, both group elements $g \in G$ and Lie algebra elements $X \in T_e(G)$ can be expressed as $N \times N$ square matrices $\mathbf{G} \in \mathbb{C}^{N \times N}$ resp $\mathbf{X} \in \mathbb{C}^{N \times N}$, where the relation between these two objects is given by the matrix exponent, i.e.

$$\mathbf{G} = \exp(\mathbf{X}) = \mathbb{I}_{N \times N} + \sum_{n=1}^{\infty} \frac{\mathbf{X}^n}{n!} = \sum_{n=0}^{\infty} \frac{\mathbf{X}^n}{n!}. \tag{2.26}$$

The advantage of representing the Lie group using matrices is that many calculations become much simpler. For instance, the Lie bracket is found by

$$[\mathbf{X}_i, \mathbf{X}_j] = \mathbf{X}_i \mathbf{X}_j - \mathbf{X}_j \mathbf{X}_i. \tag{2.27}$$

A set of Lie algebra elements $\mathbf{X}_i$, $i \in \{1, 2, \ldots, N\}$, only forms a valid N-dimensional basis for a Lie algebra if they are linearly independent and if they are closed under the Lie bracket, i.e. one can write

$$[\mathbf{X}_i, \mathbf{X}_j] = \sum_{k=1}^{N} C_{ij}^k \mathbf{X}_k \quad \forall\, i, j \in \{1, 2, \ldots, N\}, \tag{2.28}$$

where C_{ij}^k are the *structure constants*. The table of Lie brackets $[\mathbf{X}_i, \mathbf{X}_j]$, for all $i, j \in \{1, 2, \ldots, N\}$ completely describes the structure of the Lie algebra. If the exponential mapping exp is surjective, it also completely captures the structure of the Lie group. Two Lie algebras are isomorphic if the tables of Lie brackets are the same, that is, for a right choice of the Lie algebra basis $\mathbf{X}_i$.

2.6 The 2D Euclidean Motion Group

In this thesis, we particularly consider the *2D special*[1] *Euclidean motion group* $SE(2)$, and therefore we will clarify the abstract concepts described above by

[1] The word "special" indicates that mirroring is not included in the group. Further on we will always work with the special Euclidean motion group, however we will omit the word "special".

using $SE(2)$ as example. Many group theoretical concepts that are described in this thesis also apply to other Lie groups, meaning that in many places where we write "$SE(2)$", one can read "G" where G denotes an arbitrary Lie group. In Chapter 7 we apply the same concepts to the *3D special Euclidean motion group* $SE(3)$.

2.6.1 Group Properties

The 2D special Euclidean motion group $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ is the group of planar rotations ($SO(2)$) and translations ($\mathbb{R}^2$). It can be parameterized by the group elements $g = (\boldsymbol{x}, \theta)$ where $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$ are the two spatial variables that in the case of orientation scores label the domain of the image f, and θ is the orientation angle. We will use both short notation g and explicit notation $(\boldsymbol{x}, \theta)$ for group elements. The group product and group inverse of elements in $SE(2)$ are given by

$$\begin{aligned} g\,g' &= (\boldsymbol{x}, \theta)\,(\boldsymbol{x}', \theta') = (\boldsymbol{x} + \mathbf{R}_\theta \boldsymbol{x}', \theta + \theta'), \\ g^{-1} &= (-\mathbf{R}_\theta^{-1}\boldsymbol{x}, -\theta). \end{aligned} \tag{2.29}$$

The identity element is given by $e = (0, 0, 0)$. Note that the Euclidean motion group is not commutative, i.e. in general $g\,g' \neq g'\,g$. This is caused by the fact that a rotation pops up in the translation part, because of the *semi*-direct product, denoted by the symbol "$\rtimes$", between $\mathbb{R}^2$ and $SO(2)$.

The Lie algebra of $SE(2)$ is spanned by

$$T_e(SE(2)) = \operatorname{span}\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_\theta\}, \tag{2.30}$$

where $\boldsymbol{e}_x$, $\boldsymbol{e}_y$, and $\boldsymbol{e}_\theta$ are the unit vectors in the x, y, and θ direction respectively. The Lie bracket is found by (see [35])

$$[A, B] = \lim_{t \downarrow 0} t^{-2} \left(a(t)b(t)(a(t))^{-1}(b(t))^{-1} - e \right), \tag{2.31}$$

where $a, b : \mathbb{R} \to SE(2)$ are any smooth curves in $SE(2)$ with $a(0) = b(0) = e$ and $a'(0) = A$ and $b'(0) = B$. For example to find the Lie brackets for $SE(2)$ we can use for $\boldsymbol{e}_x$: $a(t) = (t, 0, 0)$, for $\boldsymbol{e}_y$: $a(t) = (0, t, 0)$, and for $\boldsymbol{e}_\theta$: $a(t) = (0, 0, t)$, yielding the following Lie brackets

$$[\boldsymbol{e}_x, \boldsymbol{e}_\theta] = -\boldsymbol{e}_y, \quad [\boldsymbol{e}_y, \boldsymbol{e}_\theta] = \boldsymbol{e}_x, \quad [\boldsymbol{e}_x, \boldsymbol{e}_y] = 0. \tag{2.32}$$

2.6.2 Representations

A few group representations of $SE(2)$ are important in this thesis. On orientation scores $U \in \mathbb{L}_2(SE(2))$ we have two different representations, defined by

$$(\mathcal{L}_g \circ U)(h) = (U \circ L_g^{-1})(h) = U(g^{-1}h), \quad g, h \in SE(2), \tag{2.33}$$

$$(\mathcal{Q}_g \circ U)(h) = (U \circ Q_g)(h) = U(h\,g), \qquad g, h \in SE(2). \tag{2.34}$$

where $\mathcal{L}_g$ is the *left-regular* representation, since the multiplication takes place on the left side, and $\mathcal{Q}_g$ is the *right-regular* representation. The symbols L_g and Q_g denote

$$\text{left-multiplication: } L_g h = g\,h, \text{ and right-multiplication: } Q_g h = h\,g. \tag{2.35}$$

The latter symbols should not be confused with $\mathcal{L}_g$ and $\mathcal{Q}_g$ as L_g and Q_g act on group elements, while $\mathcal{L}_g$ and $\mathcal{Q}_g$ act on functions on the group.

The $SE(2)$ left-regular representation on images $f \in \mathbb{L}_2(\mathbb{R}^2)$ is given by

$$(\mathcal{U}_g \circ f)(\boldsymbol{y}) = f(\mathbf{R}_\theta^{-1}(\boldsymbol{y} - \boldsymbol{x})), \quad g = (\boldsymbol{x}, \theta) \in SE(2), \quad \boldsymbol{y} \in \mathbb{R}^2. \tag{2.36}$$

Note that this representation already appeared in equation (2.4).

2.6.3 Corresponding Matrix Lie Algebra and Group

It is possible to express the Lie algebra $T_e(SE(2))$ using matrices. The Lie algebra of $SE(2)$ is spanned by the following basis

$$\mathbf{X}_x = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{X}_y = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \quad \mathbf{X}_\theta = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{2.37}$$

Consequently, the commutators are given by

$$[\mathbf{X}_x, \mathbf{X}_\theta] = -\mathbf{X}_y, \quad [\mathbf{X}_y, \mathbf{X}_\theta] = \mathbf{X}_x, \quad [\mathbf{X}_x, \mathbf{X}_y] = 0, \tag{2.38}$$

which matches the Lie brackets in (2.32).

By calculating the matrix exponents we find the following matrix representation of the $SE(2)$ group

$$\begin{aligned} \mathbf{E}_{(\boldsymbol{x},\theta)} &= \exp(x\,\mathbf{X}_x)\,\exp(y\,\mathbf{X}_y)\,\exp(\theta\,\mathbf{X}_\theta) \\ &= \begin{pmatrix} \mathbf{R}_\theta & \boldsymbol{x} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{2.39}$$

This representation can be used to apply $SE(2)$ group actions on vectors (x, y, θ). Its structure is isomorphic to the group structure defined in equation (2.29), showing that the Lie algebra matrices (2.37) are the correct ones.

2.7 Operations on Orientation Scores

In this section we will show that if we require the net operation on the image to be Euclidean invariant, we should restrict ourselves to operations Φ on orientation scores that are *left-invariant*, meaning that Φ commutes with the left-regular representation $\mathcal{L}_g$, i.e.

$$\forall g \in SE(2) : \mathcal{L}_g \circ \Phi = \Phi \circ \mathcal{L}_g. \tag{2.40}$$

On the other hand, the operation should *not* be *right-invariant*, where right-invariance means that Φ commutes with the right-regular representation

$$\forall g \in SE(2) : \mathcal{Q}_g \circ \Phi = \Phi \circ \mathcal{Q}_g. \tag{2.41}$$

Afterwards, we will discuss linear left-invariant operations, which can be expressed as $SE(2)$-*convolutions*.

2.7.1 Operations Should be Left-Invariant

For left-regular representations $\mathcal{L}_g$ it is easy to verify using equations (2.6), (2.9), (2.36), and (2.33) that the following relations hold

$$\forall g \in SE(2) : \mathcal{W}_\psi \circ \mathcal{U}_g = \mathcal{L}_g \circ \mathcal{W}_\psi \quad \text{and} \quad \forall g \in SE(2) : \mathcal{W}_\psi^* \circ \mathcal{L}_g = \mathcal{U}_g \circ \mathcal{W}_\psi^*. \tag{2.42}$$

In words, the first equation says that applying an $SE(2)$ group transformation (i.e. rotation and translation) on the image followed by an orientation score transformation is the same as applying an orientation score transformation followed by an $SE(2)$ group transformation on the orientation score. The second equation gives the corresponding relation for the inverse orientation score transformation.

By substituting $\Phi \leftarrow \mathcal{L}_g \circ \Phi$ in (2.5) and using (2.40) and (2.42), we obtain iff Φ is left-invariant

$$\begin{aligned} \mathcal{W}_\psi^* \circ \mathcal{L}_g \circ \Phi \circ \mathcal{W}_\psi &= \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{L}_g \circ \mathcal{W}_\psi, \\ \mathcal{U}_g \circ \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{W}_\psi &= \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{W}_\psi \circ \mathcal{U}_g, \\ \mathcal{U}_g \circ \Upsilon &= \Upsilon \circ \mathcal{U}_g, \end{aligned} \tag{2.43}$$

so indeed if Φ is left-invariant, the net operator Υ is Euclidean invariant and vice versa.

2.7.2 Operations Should not be Right-Invariant

For right-invariance, it is easy to verify using equations (2.6), (2.9), (2.36), and (2.34) that the following relations hold

$$\forall g \in SE(2) : \mathcal{Q}_g \circ \mathcal{W}_\psi = \mathcal{W}_{\mathcal{U}_g \circ \psi} \quad \text{and} \quad \forall g \in SE(2) : \mathcal{W}^*_{\mathcal{U}_g^{-1} \circ \psi} = \mathcal{W}_\psi^* \circ \mathcal{Q}_g. \tag{2.44}$$

Note that since $SE(2)$ is not commutative, $\mathcal{W}_{\mathcal{U}_g \circ \psi} \neq \mathcal{W}_\psi \circ \mathcal{U}_g$, which can be easily seen as follows

$$\begin{aligned}(\mathcal{W}_{\mathcal{U}_g \circ \psi} \circ f)(h) &= (\mathcal{U}_h \circ \mathcal{U}_g \circ \psi, f)_{\mathbb{L}_2(\mathbb{R}^2)} \\ &\neq (\mathcal{U}_g \circ \mathcal{U}_h \circ \psi, f)_{\mathbb{L}_2(\mathbb{R}^2)} = (\mathcal{W}_\psi \circ \mathcal{U}_g \circ f)(h).\end{aligned} \tag{2.45}$$

In other words, we see that the group transformation $\mathcal{U}_g$ on the kernel ψ cannot be expressed as a group transformation on the image f since $\mathcal{U}_g \circ \mathcal{U}_h \neq \mathcal{U}_h \circ \mathcal{U}_g$.

By substituting $\Phi \leftarrow \mathcal{Q}_g \circ \Phi$ in (2.5) and using (2.41) and (2.44), we obtain

$$\begin{aligned}\mathcal{W}_\psi^* \circ \mathcal{Q}_g \circ \Phi \circ \mathcal{W}_\psi &= \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{Q}_g \circ \mathcal{W}_\psi, \\ \mathcal{W}_{\mathcal{U}_g^{-1}\psi}^* \circ \Phi \circ \mathcal{W}_\psi &= \mathcal{W}_\psi^* \circ \Phi \circ \mathcal{W}_{\mathcal{U}_g \psi}.\end{aligned} \tag{2.46}$$

This equation only matches the requirement (2.4) of Euclidean invariance of Υ if $\psi = \mathcal{U}_g \circ \psi$ for all $g \in SE(2)$. This is not a sensible requirement on ψ, as the only solution is $\psi = 0$. Clearly, *right-invariance is prohibited.*

2.7.3 Left-Invariant Linear Operations are Group Convolutions

All bounded linear operators $\Phi : \mathbb{L}_2(SE(2)) \to \mathbb{L}_\infty(SE(2))$ are kernel operators (cf. the Dunford-Pettis theorem [33, page 21]), that is, there exists a kernel K such that

$$(\Phi(U))(g) = \int_{SE(2)} K(g,h)U(h)\mathrm{d}h, \quad g,h \in SE(2), \tag{2.47}$$

where $K : SE(2) \times SE(2) \to \mathbb{C}$ fulfills

$$\sup_g \left(\int_{SE(2)} |K(g,h)|^2 dh \right)^{1/2} < \infty. \tag{2.48}$$

If we require Φ to be left-invariant, i.e. $\Phi \circ \mathcal{L}_g \circ U = \mathcal{L}_g \circ \Phi \circ U$, we obtain using (2.47)

$$\begin{aligned}(\Phi \circ \mathcal{L}_p \circ U)(g) &= \int_{SE(2)} K(g,h)(\mathcal{L}_p \circ U)(h) dh \\ &= \int_{SE(2)} K(g,h)U(p^{-1}h) dh \quad \text{(substitute } h \leftarrow p\,h\text{)} \\ &= \int_{SE(2)} K(g,ph)U(h) dh,\end{aligned} \tag{2.49}$$

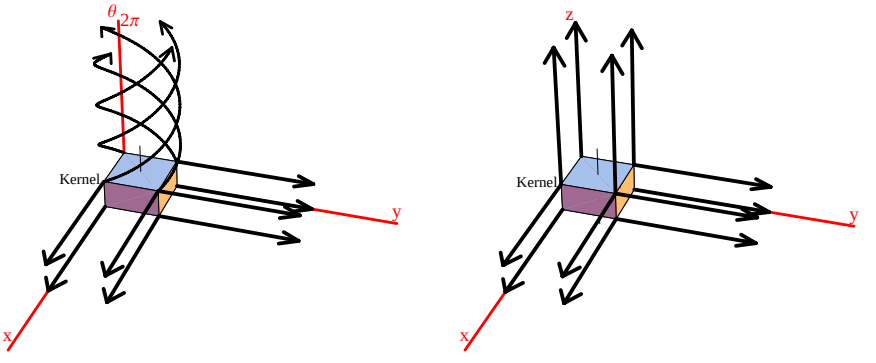


Figure 2.4: Schematic view of a normal $SE(2)$-convolution (left) and an $\mathbb{R}^3$-convolution (right). Any convolution $K * f$ can be envisioned by the kernel K "moving" over the entire domain of f where the result of the convolution is found by taking inner products at all positions. The kernel is indicated as a box, and the 3 axes denote the orientation score domain and 3D image domain respectively. As indicated by the arrows, in both cases the kernel is moving over the x, y-plane. In the case of the $SE(2)$-convolution, there is an additional twist when moving in the orientation dimension.

and

$$(\mathcal{L}_p \circ \Phi \circ U)(g) = \int_{SE(2)} K(p^{-1}g, h)U(h)dh. \tag{2.50}$$

In order to make the right-hand sides of (2.49) and (2.50) equal, $K(g, ph) = K(p^{-1}g, h)$ must hold $\forall\, p, g, h \in SE(2)$. If we replace g by pg we obtain $K(pg, ph) = K(g, h)$, and we see that K should not change if we left-multiply both arguments of K with an arbitrary group element p. So $K(g, h) = K(e, g^{-1}h)$, whence if we define a new function $\Psi : SE(2) \to \mathbb{C}$ such that

$$\Psi(g) = K(e, g^{-1}), \tag{2.51}$$

and then use (2.47) and (2.51), we obtain the $SE(2)$-*convolution*

$$(\Psi *_{SE(2)} U)(g) = \int_{SE(2)} \Psi(h^{-1}g)U(h)dh. \tag{2.52}$$

Explicitly,

$$(\Psi *_{SE(2)} U)(\boldsymbol{x}, \theta) = \int_{\mathbb{R}^2} \int_0^{2\pi} \Psi(\mathbf{R}_{\theta'}^{-1}(\boldsymbol{x} - \boldsymbol{x}'), \theta - \theta')\, U(\boldsymbol{x}', \theta')\, \mathrm{d}\theta'\, \mathrm{d}\boldsymbol{x}'. \tag{2.53}$$

Figure 2.4 schematically shows how an $SE(2)$-convolution can be envisioned. Note that equation (2.52) is in fact a straightforward generalization of $\mathbb{R}^n$-convolution

to any other Lie group, by replacing $SE(2)$ by any other group. That means that we obtain the $\mathbb{R}^n$-convolution if we use the translation group $\mathbb{R}^n$ in (2.52), i.e.

$$(f *_{\mathbb{R}^n} g)(\boldsymbol{x}) = \int_{\mathbb{R}^n} f(\boldsymbol{x} - \boldsymbol{x}')\, g(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'. \tag{2.54}$$

In Chapter 3, the $SE(2)$-convolution will be described in more detail, and several methods to implement the $SE(2)$-convolution will be proposed.

2.8 Differential Geometry in Orientation Scores

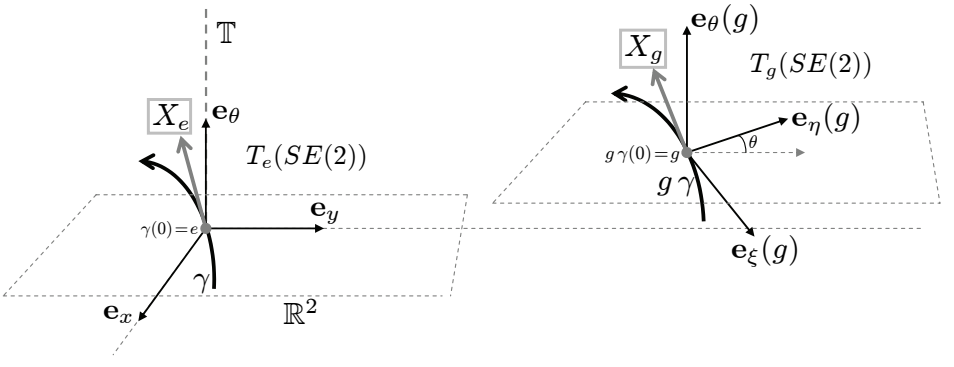
To perform analysis of the local structure in orientation scores, for instance to estimate whether locally an oriented structure is present or not, *differential geometry* is an essential tool. Therefore, we will establish the necessary basics on differential geometry in $SE(2)$. First, we will define left-invariant tangent vectors and vector fields. Then we will introduce left-invariant differential operators. Both tangent vectors and differential operators will also help to get an intuitive view on the concept of left-invariance, which will be graphically illustrated.

Subsequently, we will explain the difference between tangent spaces and dual tangent spaces in $SE(2)$, and between vectors and covectors, which is a relevant distinction to be made in this case. Then we come to a definition of a left-invariant inner product and norm on tangent spaces.

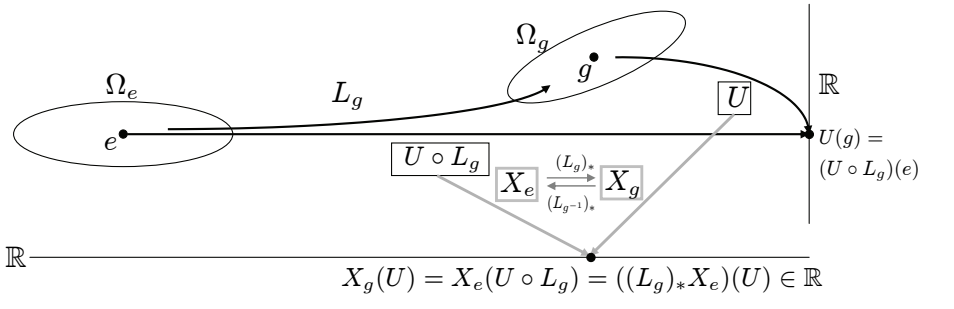
Finally the introduced differential geometric concepts will be used to introduce the notion of horizontality, we will define exponential curves, and finally we will introduce the features *curvature* and *deviation from horizontality*.

2.8.1 Left-invariant Tangent Vectors and Vector Fields

In Figure 2.5(a), at the left side, a curve $\gamma : \mathbb{R} \to SE(2)$ is shown that is tangent to tangent vector $X_e = c^{\xi}\boldsymbol{e}_x + c^{\eta}\boldsymbol{e}_y + c^{\theta}\boldsymbol{e}_\theta \in T_e(SE(2))$ at the unity element $e \in SE(2)$. If we left-multiply curve γ by $g = (\boldsymbol{x}, \theta) \in SE(2)$ the original curve γ is translated over $\boldsymbol{x}$ and rotated over θ, yielding the curve $g\,\gamma$ shown on the right side of the figure. At position g, the curve $g\,\gamma$ has tangent vector $X_g \in T_g(SE(2))$, which corresponds to the tangent vector $X_e \in T_e(SE(2))$ of the original curve γ, in the sense that X_e is "transported in a left-invariant way" to tangent vector X_g. The operation of transporting tangent vectors from $T_e(SE(2))$ to $T_g(SE(2))$ is called the *push-forward* of left-multiplication and is denoted by $X_g = (L_g)_* X_e$. Intuitively, the push-forward operator allows to move tangent vectors to tangent spaces at different group elements.



(a) (a) Left-invariance of tangent vectors to curves



(b) (b) Left-invariance of tangent vectors considered as differential operators

Figure 2.5: Illustrations of the concept of left-invariance, from two different perspectives: (a) considered as tangent vectors tangent to curves, i.e. $X_g = c^\xi \boldsymbol{e}_\xi(g) + c^\eta \boldsymbol{e}_\eta(g) + c^\theta \boldsymbol{e}_\theta(g)$ for all $g \in SE(2)$, and (b) considered as differential operators on a locally defined smooth function U, i.e. $X_g = c^\xi \partial_\xi\big|_g + c^\eta \partial_\eta\big|_g + c^\theta \partial_\theta\big|_g$ for all $g \in SE(2)$. The push forward $(L_g)_* : T_e(SE(2)) \to T_g(SE(2))$ connects the tangent space at the unity element $T_e(SE(2))$ to all tangent spaces $T_g(SE(2))$ in "a left-invariant way". See text (Subsections 2.8.1 and 2.8.2) for details.

With the push-forward we can introduce the notion of a *left-invariant vector field*, which is a tangent vector field $SE(2) \to T_g(SE(2))$ that fulfills the property $X_g = (L_g)_* X_e$ for all $g \in SE(2)$ and a fixed $X_e \in T_e(SE(2))$.

The Lie algebra $T_e(SE(2))$ is spanned by the basis $\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_\theta\}$, as is shown in Figure 2.5(a) at the left side. The same basis can also be used as standard basis for $T_g(SE(2))$ for all g, however, it is more natural to introduce a different set of basis vectors by constructing left-invariant vector fields by push-forward of the three basis vectors $\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_\theta\}$, yielding

$$\begin{aligned} \{\boldsymbol{e}_\xi(g), \boldsymbol{e}_\eta(g), \boldsymbol{e}_\theta(g)\} &= \{(L_g)_* \boldsymbol{e}_x, (L_g)_* \boldsymbol{e}_y, (L_g)_* \boldsymbol{e}_\theta\} \\ &= \{\cos\theta\, \boldsymbol{e}_x + \sin\theta\, \boldsymbol{e}_y, -\sin\theta\, \boldsymbol{e}_x + \cos\theta\, \boldsymbol{e}_y, \boldsymbol{e}_\theta\}, \end{aligned} \tag{2.55}$$

where $\text{span}\{\boldsymbol{e}_\xi(g), \boldsymbol{e}_\eta(g), \boldsymbol{e}_\theta(g)\} = T_g(SE(2))$. These basis vectors are shown at the right side of Figure 2.5(a). The basis has the useful property that $X_g = c^\xi \boldsymbol{e}_\xi(g) + c^\eta \boldsymbol{e}_\eta(g) + c^\theta \boldsymbol{e}_\theta(g)$ for any g if X_g is obtained by push-forward $X_g = (L_g)_* X_e$. So, if two tangent vectors at different positions in $SE(2)$ relate to each other by the push-forward, they have the same vector components $(c_\xi, c_\eta, c_\theta)$. For notational simplicity the dependency on g is usually omitted further on, but it is important to realize that $\boldsymbol{e}_\xi$ and $\boldsymbol{e}_\eta$ do depend on θ.

2.8.2 Left-invariant Differential Operators

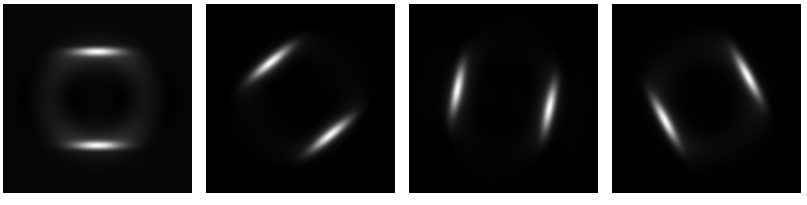
In Figure 2.5(b) it is shown how X_e and X_g can also be viewed as differential operators, acting on a function $U : SE(2) \to \mathbb{R}$. If we are using X_e and X_g in the context of differential operators we will replace all occurrences of $\boldsymbol{e}_i$ by ∂_i, which is short-hand notation for $\frac{\partial}{\partial_i}$. In Figure 2.5(b), the codomain of U is the vertical $\mathbb{R}$-axis. X_g can be viewed as an operator that calculates the derivative of U at g in the direction specified by X_g, i.e.

$$\begin{aligned} X_g(U) &= (c^\xi \partial_\xi\big|_g + c^\eta \partial_\eta\big|_g + c^\theta \partial_\theta\big|_g) U \\ &= \big(c^\xi(\cos\theta\, \partial_x + \sin\theta\, \partial_y) + c^\eta(-\sin\theta\, \partial_x + \cos\theta\, \partial_y) + c^\theta \partial_\theta\big)\, U, \end{aligned} \tag{2.56}$$

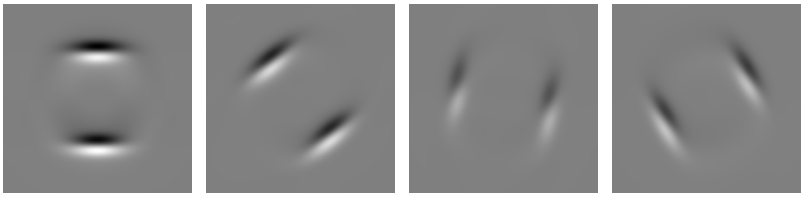
yielding a scalar on the horizontal $\mathbb{R}$-axis on the bottom of the figure. The same result can also be obtained by first translating and rotating U over g, i.e. $\mathcal{L}_g(U) = U \circ L_g$, such that the original neighborhood Ω_g is shifted to neighborhood Ω_e,

$$X_g(U) = X_e(U \circ L_g) = (c^\xi \partial_x + c^\eta \partial_y + c^\theta \partial_\theta)(U \circ L_g). \tag{2.57}$$

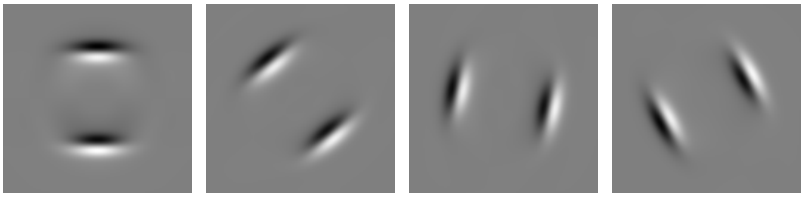
The set of differential operators $\{\partial_\xi, \partial_\eta, \partial_\theta\}$ is the appropriate set of differential operators to be used in orientation scores instead of the set $\{\partial_x, \partial_y, \partial_\theta\}$ (which also seems a logical choice because of the sampling axes), since it has the following advantages



(a) Orientation score of circle image



(b) Cartesian derivative ∂_y



(c) Left-invariant derivative ∂_η

Figure 2.6: The difference between left-invariant derivatives and Cartesian derivatives, showed on an orientation score of an image with a single circle, cf. Figure 1.4 on page 6. From left to right, several different orientations are shown. Comparing the derivatives ∂_y and ∂_η we observe that ∂_η is invariant under the orientation, i.e. the interpretation of ∂_η stays te same.

- Since all 3 differential operators $\{\partial_\xi, \partial_\eta, \partial_\theta\}$ are left-invariant, we can be sure that all $SE(2)$-coordinate independent linear and nonlinear combinations of these operators are also left-invariant.
- They have a clear interpretation, since ∂_ξ is always the spatial derivative tangent to the orientation θ and ∂_η is always orthogonal to this orientation. Figure 2.6 illustrates this for ∂_η versus ∂_y.

When constructing higher order left-invariant derivatives, it is important to note that the order of applying left-invariant derivatives matters, i.e. not all the left-invariant derivatives $\{\partial_\xi, \partial_\eta, \partial_\theta\}$ commute. The nonzero commutators are given by

$$\begin{aligned}[\partial_\theta, \partial_\xi] &= \partial_\theta(\cos\theta\partial_x + \sin\theta\partial_y) - (\cos\theta\partial_x + \sin\theta\partial_y)\partial_\theta = \partial_\eta,\\ [\partial_\theta, \partial_\eta] &= \partial_\theta(-\sin\theta\partial_x + \cos\theta\partial_y) - (-\sin\theta\partial_x + \cos\theta\partial_y)\partial_\theta = -\partial_\xi.\end{aligned} \tag{2.58}$$

We observe that the differential structure on any position g is isomorphic to the Lie algebra structure defined in equation (2.32). This follows from the fact that the position of the identity element e in an orientation score U is an arbitrary choice, and we can easily change the position of e by the group transformation $U \circ L_g$, as is shown in Figure 2.5(b) and (2.57).

2.8.3 Tangent Spaces and Dual Tangent Spaces

In Subsection 2.8.1 we introduced the tangent space $T_g(SE(2))$. An element of $T_g(SE(2))$ is a *vector* which is denoted in a basis-independent way by $c^\xi \partial_\xi\big|_g + c^\eta \partial_\eta\big|_g + c^\theta \partial_\theta\big|_g \in T_g(SE(2))$, where $(c^\xi, c^\eta, c^\theta) \in \mathbb{R}^3$ are the basis-dependent vector components and $\{\partial_\xi\big|_g, \partial_\eta\big|_g, \partial_\theta\big|_g\}$ is our left-invariant standard basis. We will further on work with the vector components and use the notation $\boldsymbol{c} = \left(c^\xi, c^\eta, c^\theta\right)^{\mathrm{T}}$. Note that the physical dimensions of these vector components are (length, length, 1) respectively.

Similarly a *covector* is denoted by $c_\xi \mathrm{d}\xi\big|_g + c_\eta \mathrm{d}\eta\big|_g + c_\theta \mathrm{d}\theta\big|_g \in T_g^*(SE(2))$, where $\mathrm{d}\theta$, $\mathrm{d}\xi$, and $\mathrm{d}\eta$ span the basis of the *dual* tangent space $T_g^*(SE(2))$. The relation between the tangent space and the dual tangent space is established by the Kronecker product

$$\langle \mathrm{d}p\big|_g, \partial_q\big|_g \rangle = \delta_{pq} \quad \text{with } p, q \in \{\xi, \eta, \theta\}, \tag{2.59}$$

so for example $\langle \mathrm{d}\theta, \partial_\xi \rangle = 0$ and $\langle \mathrm{d}\theta, \partial_\theta \rangle = 1$. For the basis-dependent covector components we use the notation $\hat{\boldsymbol{b}} = (b_\xi, b_\eta, b_\theta)$ where the "hat" in the notation of $\hat{\boldsymbol{b}}$ allows us to distinguish between vectors and covectors. The physical dimensions of the covector components are (1/length, 1/length, 1) respectively.

The Kronecker product on covector components $\hat{\boldsymbol{b}}$ and vector components $\boldsymbol{c}$ is defined by

$$\langle \hat{\boldsymbol{b}}, \boldsymbol{c} \rangle = b_\xi c^\xi + b_\eta c^\eta + b_\theta c^\theta, \tag{2.60}$$

where the resulting number is dimensionless.

An example of a *vector* in $T_g(SE(2))$ is given by the tangent vector of a curve $q : \mathbb{R} \to SE(2)$, i.e.

$$\dot{q}(t) = \dot{x}(t)\,\partial_x + \dot{y}(t)\,\partial_y + \dot{\theta}(t)\,\partial_\theta, \quad \dot{q}(t) \in T_{q(t)}(SE(2)), \tag{2.61}$$

Note that we have ∂_x and ∂_y instead of ∂_ξ and ∂_η since we use our standard parametrization for a group element (x, y, θ) as defined in Subsection 2.6.1, instead of (ξ, η, θ).

An example of a *covector* in $T_g^*(SE(2))$ is the left-invariant gradient operator on an orientation score, which is given by

$$dU = \frac{\partial U}{\partial \xi} \mathrm{d}\xi + \frac{\partial U}{\partial \eta} \mathrm{d}\eta + \frac{\partial U}{\partial \theta} \mathrm{d}\theta. \tag{2.62}$$

The covector components of the gradient are obtained by the well-known nabla operator

$$\nabla U = \left(\frac{\partial U}{\partial \xi}, \frac{\partial U}{\partial \eta}, \frac{\partial U}{\partial \theta} \right)^{\mathrm{T}}. \tag{2.63}$$

2.8.4 Definition of an Inner Product and Norm

The inner product between two vectors $\boldsymbol{c}, \boldsymbol{b} \in \mathbb{R}^n$ is usually defined as

$$(\boldsymbol{c}, \boldsymbol{b}) = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta_{ij} c^i b^j, \tag{2.64}$$

where δ_{ij} is the Kronecker delta function ($\delta_{ij} = 1$ if $i = j$ and 0 otherwise). In the case of $T_g(SE(2))$ we could simply use the same inner product, however, we add up quantities with different physical dimensions since the components c^ξ, c^η, and c^θ do not have the same physical dimensions. Therefore, we introduce a parameter μ with physical dimension 1/length and define as inner product

$$(\boldsymbol{c}, \boldsymbol{b})_\mu = \mu^2 c^\xi b^\xi + \mu^2 c^\eta b^\eta + c^\theta b^\theta. \tag{2.65}$$

The parameter μ ensures that all components of the inner product are dimensionless. The value of the parameter determines how the distance in the spatial dimensions relates to distance in the orientation dimension.

From inner product (2.65) we can calculate the Grammian matrix

$$\mathbf{G}_\mu = \begin{pmatrix} (\partial_\xi, \partial_\xi)_\mu & (\partial_\xi, \partial_\eta)_\mu & (\partial_\xi, \partial_\theta)_\mu \\ (\partial_\eta, \partial_\xi)_\mu & (\partial_\eta, \partial_\eta)_\mu & (\partial_\eta, \partial_\theta)_\mu \\ (\partial_\theta, \partial_\xi)_\mu & (\partial_\theta, \partial_\eta)_\mu & (\partial_\theta, \partial_\theta)_\mu \end{pmatrix} = \begin{pmatrix} \mu^2 & 0 & 0 \\ 0 & \mu^2 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.66}$$

The Grammian matrix establishes the relation between the components of vectors and covectors by $\hat{\boldsymbol{c}} = \mathbf{G}_\mu \boldsymbol{c}$ and between the inner product and Kronecker product

$$(\boldsymbol{c}, \boldsymbol{b})_\mu = \langle \mathbf{G}_\mu \boldsymbol{c}, \boldsymbol{b} \rangle = \langle \hat{\boldsymbol{c}}, \boldsymbol{b} \rangle. \tag{2.67}$$

Consequently, the inner product between two covectors is given by

$$(\hat{\boldsymbol{c}}, \hat{\boldsymbol{b}})_\mu = \langle \hat{\boldsymbol{c}}, \mathbf{G}_\mu^{-1} \hat{\boldsymbol{b}} \rangle = \mu^{-2} c_\xi b_\xi + \mu^{-2} c_\eta b_\eta + c_\theta b_\theta. \tag{2.68}$$

From the inner product on $T_g(SE(2))$ we can now induce a norm on vectors and covectors in the regular way, i.e. by

$$\|\boldsymbol{c}\|_\mu = \sqrt{(\boldsymbol{c}, \boldsymbol{c})_\mu} \quad \text{and} \quad \|\hat{\boldsymbol{c}}\|_\mu = \sqrt{(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}})_\mu}. \tag{2.69}$$

This inner product is left-invariant, which directly follows from the fact that we express the inner product using vector components of the left-invariant standard basis. However, the inner product is *not* right-invariant, which is not a problem since right-invariance is not desirable in our framework, as explained in Subsection 2.7.2. For a more theoretical motivation of the inner product and its properties see [40].

2.8.5 Horizontality

A curve $q : \mathbb{R} \to SE(2)$ in the orientation score, denoted by its components as $q(t) = (x(t), y(t), \theta(t))$, is *horizontal* at $t \in \mathbb{R}$ iff

$$\theta(t) = \angle \left(\frac{\mathrm{d}x(t)}{\mathrm{d}t}, \frac{\mathrm{d}y(t)}{\mathrm{d}t} \right), \tag{2.70}$$

where $\angle(x, y) = \arg(x + i\, y)$. In words, horizontal curves have the property that the direction of the curve $\mathbb{P}_{\mathbb{R}^2} q$, i.e. the curve projected to the spatial plane $\mathbb{R}^2$, coincides with the orientation θ of the curve in $SE(2)$. Therefore all tangent vectors over the curve do not contain an $\boldsymbol{e}_\eta$ component, i.e. an equivalent formulation for horizontality of q is

$$\left(\frac{\mathrm{d}q(t)}{\mathrm{d}t}, \boldsymbol{e}_\eta(q(t)) \right)_\mu = \mu^2 \left(-\sin\theta \frac{\mathrm{d}x(t)}{\mathrm{d}t} + \cos\theta \frac{\mathrm{d}y(t)}{\mathrm{d}t} \right) = 0, \quad \forall\, t \in \mathbb{R}. \tag{2.71}$$

On a horizontal curve, ∂_ξ is always the spatial derivative tangent to the curve and ∂_η is always orthogonal to the curve.

A curve $(x(t), y(t))$ in an image would render a perfectly horizontal response, if at each spatial position along the curve the response in the orientation dimension would be a δ-spike, i.e. $U_f(x(t), y(t), \theta) = \delta(\arg(\frac{\mathrm{d}x(t)}{\mathrm{d}t} + i\,\frac{\mathrm{d}y(t)}{\mathrm{d}t}) - \theta)$. In that case we could say that the curve renders an exactly *horizontal response* in the orientation score. However, by construction, cf. Section 2.4, the response in the orientation score resulting from a curve in the corresponding image always exhibits some uncertainty in orientation. Therefore, in practice, the orientation score transformation cf. Section 2.4 renders orientation scores in which image curves render *approximately* horizontal responses.

2.8.6 Exponential Curves

An *exponential curve* is a curve $\gamma_{\boldsymbol{c}} : \mathbb{R} \to SE(2)$ for which the components of the tangent vector expressed in the left-invariant basis $\{\boldsymbol{e}_\xi, \boldsymbol{e}_\eta, \boldsymbol{e}_\theta\}$ are constant over the entire parametrization (if necessary after reparametrization of t), i.e.

$$\frac{\mathrm{d}}{\mathrm{d}t}\gamma_{\boldsymbol{c}}(t) = c^\xi \boldsymbol{e}_\xi(\gamma_{\boldsymbol{c}}(t)) + c^\eta \boldsymbol{e}_\eta(\gamma_{\boldsymbol{c}}(t)) + c^\theta \boldsymbol{e}_\theta(\gamma_{\boldsymbol{c}}(t)), \quad \text{for all } t \in \mathbb{R}. \tag{2.72}$$

These curves are analogous to *straight* lines in $\mathbb{R}^n$, which also have a constant tangent vector relative to the cartesian basis $\{\boldsymbol{e}_x, \boldsymbol{e}_y\}$.

An exponential curve is obtained by an exponential mapping of Lie algebra elements, which explains why they are called "exponential". That is, an exponential curve passing through the identity element $e \in SE(2)$ at $t = 0$ can be written as

$$\begin{aligned} \gamma_{\boldsymbol{c}}(t) &= \exp\left(t\left(c^\xi \partial_\xi\big|_{g=e} + c^\eta \partial_\eta\big|_{g=e} + c^\theta \partial_\theta\big|_{g=e}\right)\right) \\ &= \exp\left(t(c^\xi \partial_x + c^\eta \partial_y + c^\theta \partial_\theta)\right), \end{aligned} \tag{2.73}$$

and an exponential curve passing through point $g_0 \in SE(2)$ can be obtained by left-multiplication with g_0, i.e. $g_0\,\gamma_{\boldsymbol{c}}(t)$. The easiest way to calculate exponential curves is by using the matrix Lie algebra elements in equation (2.37) and the matrix exponent as defined in (2.26), i.e. $\mathbf{G}(t) = \mathbf{E}_{g_0} \exp\left(t(c^\xi \mathbf{X}_x + c^\eta \mathbf{X}_y + c^\theta \mathbf{X}_\theta)\right)$. For the case $c^\theta \neq 0$ this results in

$$\begin{aligned} g_0\,\gamma_{\boldsymbol{c}}(t) &= \begin{pmatrix} x_0 + \frac{c^\xi}{c^\theta}\,\mu(tc^\theta, \theta_0) + \frac{c^\eta}{c^\theta}\,\nu(tc^\theta, \theta_0) \\ y_0 + \frac{c^\xi}{c^\theta}\,\nu(tc^\theta, \theta_0) + \frac{c^\eta}{c^\theta}\,\mu(tc^\theta, \theta_0) \\ tc^\theta + \theta_0 \end{pmatrix}, \\ \text{with} \quad \mu(tc^\theta, \theta_0) &= \sin(tc^\theta + \theta_0) - \sin(\theta_0), \\ \nu(tc^\theta, \theta_0) &= \cos(tc^\theta + \theta_0) - \cos(\theta_0). \end{aligned} \tag{2.74}$$

These exponential curves represent *spirals* in $SE(2)$. The exponential curves for

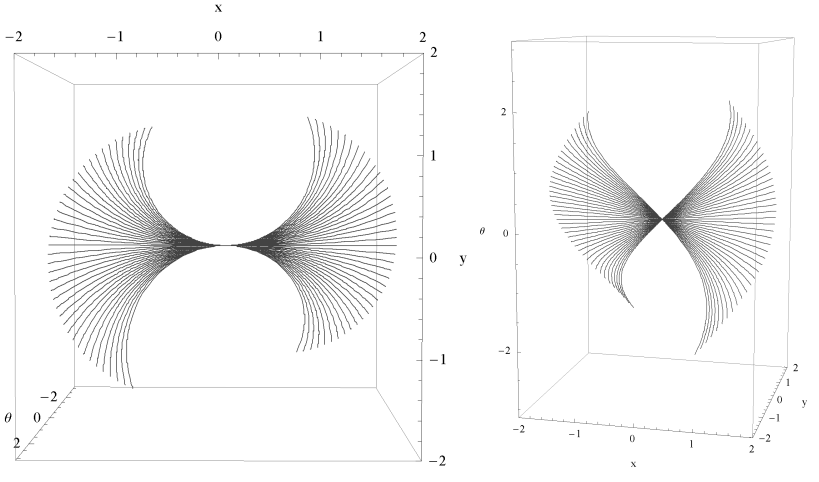


Figure 2.7: Horizontal exponential curves in $SE(2)$ for a range of different curvature values, shown from two different perspectives. The left image shows that these curves are circular arcs when projected onto the spatial plane.

the special case $c^\theta = 0$ are

$$g_0\,\gamma_{\boldsymbol{c}}(t) = \begin{pmatrix} x_0 + tc^\xi \cos(\theta_0) - t\,c^\eta \sin(\theta_0) \\ y_0 + tc^\xi \sin(\theta_0) - t\,c^\eta \cos(\theta_0) \\ \theta_0 \end{pmatrix}, \tag{2.75}$$

which are straight lines in $SE(2)$ with constant orientation θ.

Horizontal exponential curves are exponential curves that are horizontal cf. (2.70) for all $t \in \mathbb{R}$. They form the subset of all exponential curves with $c^\eta = 0$, see Figure 2.7.

2.8.7 Curvature and Deviation from Horizontality

We define two features of exponential curves with a clear geometrical interpretation: *curvature* and *deviation from horizontality*. These features can be expressed using the tangent vector components $\boldsymbol{c} = (c^\xi, c^\eta, c^\theta)$ of an exponential curve $\gamma_{\boldsymbol{c}}$, as is illustrated in Figure 2.8.

The *curvature* of an exponential curve in $SE(2)$ that is projected onto $\mathbb{R}^2$ is given

by

$$\begin{aligned}\boldsymbol{\kappa}(t) &= \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbb{P}_{\mathbb{R}^2}\gamma_{\mathbf{c}}(t)) \\ &= \frac{c^\theta}{(c^\eta)^2+(c^\xi)^2}\begin{pmatrix} -c^\eta\cos(t\,c^\theta+\theta_0) - c^\xi\sin(t\,c^\theta+\theta_0) \\ c^\xi\cos(t\,c^\theta+\theta_0) - c^\eta\sin(t\,c^\theta+\theta_0)\end{pmatrix},\end{aligned} \tag{2.76}$$

where t must be the arc length parametrization in the spatial plane, so $||\frac{\mathrm{d}}{\mathrm{d}t}(\mathbb{P}_{\mathbb{R}^2}\gamma_{\mathbf{c}}(t))|| = 1$. The signed norm of the curvature vector, which is independent on t, is

$$\kappa = \|\boldsymbol{\kappa}(t)\|\,\mathrm{sign}(\boldsymbol{\kappa}(t)\cdot\boldsymbol{e}_\eta) = \frac{c^\theta\,\mathrm{sign}(c^\xi)}{\sqrt{(c^\eta)^2+(c^\xi)^2}}. \tag{2.77}$$

This equation shows that the curvature is equal to the slope at which the curve in the orientation score meets the spatial plane, see Figure 2.8. Note that if $c^\eta = c^\xi = 0$, the limit of (2.76) and (2.77) goes to infinity; this makes sense since an exponential curve maps to a point in the image plane, and a point can be considered as a circle with radius 0.

For a *horizontal* exponential curve we know that $c^\eta = 0$ and the curvature expression simplifies to

$$\kappa = \frac{c^\theta}{c^\xi}. \tag{2.78}$$

The curvature κ fully describes a horizontal exponential curve $\gamma_{\mathbf{c}}(t)\big|_{c^\eta=0}$. For a non-horizontal exponential curve, we also need the *deviation from horizontality* d_{H} which is for $t=0$ given by

$$d_{\mathrm{H}} = \arctan\left(\frac{c^\eta}{c^\xi}\right), \tag{2.79}$$

i.e. d_{H} is the angle between $\boldsymbol{e}_\xi$ and the tangent vector of the exponential curve at $t=0$ projected onto $\mathbb{R}^2$, as shown in Figure 2.8.

2.9 Left-invariant Evolution Equations on $SE(2)$

In this section we introduce left-invariant evolution equations. First, we introduce the diffusion equation on $SE(2)$. Later in this thesis we will use the linear diffusion equation for regularization of orientation scores (Chapter 5) and the nonlinear one will be used for enhancement of elongated structures (Chapter 6). Second, we will add convection to the evolution equation, which will prove to be useful for curve *completion*, i.e. filling gaps within curves, in Section 3.8. Finally, we will briefly address the question whether these evolution equations constitute *scale spaces* on $SE(2)$.

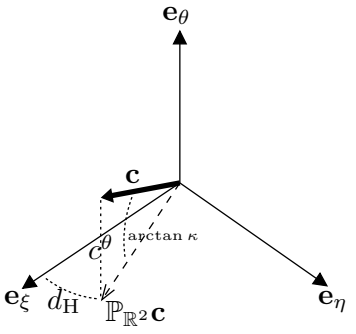


Figure 2.8: Illustration of curvature and deviation from horizontality (Subsection 2.8.7). Note that for visualization reasons, the lengths of the vectors are arbitrary. The true lengths are given by $\|\boldsymbol{c}\|_\mu = \|\boldsymbol{e}_\theta\|_\mu = \|\boldsymbol{e}_\xi\|_\mu = \|\boldsymbol{e}_\eta\|_\mu = 1$ and $\|\partial_a\|_\mu = \|\partial_b\|_\mu = \|\partial_c\|_\mu = \mu$.

2.9.1 The Diffusion Equation on $SE(2)$

The linear diffusion equation on an n-dimensional function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\begin{cases} \partial_t u(\boldsymbol{x}, t) = \nabla \cdot \mathbf{D}\, \nabla u(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \mathbb{R}^n, \quad t \geq 0, \\ u(\boldsymbol{x}, t = 0) = f(\boldsymbol{x}), \end{cases} \tag{2.80}$$

where the ∇ operator is the gradient with respect to the spatial coordinates, and the constant diffusion tensor $\mathbf{D}$ is a positive definite matrix of size $n \times n$. This diffusion equation is left-invariant with respect to the translation group, which in this case means translation invariance cf. (2.1) where $\mathcal{T}$ is the representation of the translation group on $\mathbb{L}_2(\mathbb{R}^n)$.

In the same way, we construct the left-invariant diffusion equation on $SE(2)$.

$$\begin{cases} \partial_t W(g,t) = \nabla \cdot \mathbf{D}\, \nabla W(g,t) \\ \qquad\qquad = \begin{pmatrix} \partial_\xi & \partial_\eta & \partial_\theta \end{pmatrix} \begin{pmatrix} D_{\xi\xi} & D_{\xi\eta} & D_{\xi\theta} \\ D_{\eta\xi} & D_{\eta\eta} & D_{\eta\theta} \\ D_{\theta\xi} & D_{\theta\eta} & D_{\theta\theta} \end{pmatrix} \begin{pmatrix} \partial_\xi \\ \partial_\eta \\ \partial_\theta \end{pmatrix} W(g,t), \\ W(g, t = 0) = U_f(g), \end{cases} \tag{2.81}$$

where $g \in SE(2)$, $t \geq 0$, the gradient ∇ is defined in (2.63), and the diffusion tensor $\mathbf{D}$ is a positive (semi)definite[2] 3×3 matrix.

[2] In $\mathbb{R}^n$ the diffusion tensor must be positive definite to ensure smoothing in all directions. On $SE(2)$ a positive semidefinite diffusions tensor does already render a smooth result in all directions as long as at least one term $D_{3i} = D_{i3} > 0$ for $i \in \{1, 2, 3\}$ and at least one term $D_{ji} = D_{ij} > 0$ for $i, j \in \{1, 2\}$. In that case, due to the nonzero commutators, the diffusion in

The solution of equation (2.81) can be written as

$$W(\cdot, t) = \exp(t(\nabla \cdot \mathbf{D}\,\nabla))U_f, \tag{2.82}$$

where the exponent for operators is defined as

$$\exp(t\mathcal{A})U_f = \sum_{j=0}^{\infty} = \frac{t^j}{j!}(\mathcal{A})^j\, U_f. \tag{2.83}$$

We distinguish different types of diffusion processes. A diffusion process is defined to be

- *Linear and constant*, if diffusion tensor $\mathbf{D} = [D_{ij}]$ is constant;
- *Linear and adaptive*, if $\mathbf{D}$ is a function on $SE(2)$ that is calculated from the initial condition U_f, i.e. we substitute $\mathbf{D} \leftarrow \mathbf{D}(U_f)(g)$ in (2.81);
- *Nonlinear and adaptive*, if $\mathbf{D}$ is a function on $SE(2)$ that is calculated from the evolving W, i.e. we substitute $\mathbf{D} \leftarrow \mathbf{D}(W(\cdot, t))(g)$ in (2.81).

In the linear case where $\mathbf{D}$ is constant, the diffusion process can be calculated by using an $SE(2)$-convolution (Subsection 2.7.3) where the kernel Ψ is the Green's function of the diffusion process, that is $\Psi_t(g) = \exp(t\,\nabla \cdot \mathbf{D}\,\nabla)\delta(g)$ where δ is the Dirac delta distribution. Compared to diffusion on $\mathbb{R}^n$, where the Green's function is simply a Gaussian, the Green's function of diffusion on $SE(2)$ is complicated. In [39] exact expressions and practical approximations for this Green's function are derived.

In Chapter 6 we will also apply nonlinear adaptive diffusion for the purpose of curve enhancement. In that case the diffusion equation has to be implemented using an iterative numerical scheme. Next we will consider two interesting special cases of the diffusion equation (2.81).

2.9.1.1 Special Case: μ-Isotropic Diffusion

There is no inherent notion of isotropy in $SE(2)$. We can, however, define an artificial (but practically useful, as we will see in Chapter 5) notion of *μ-isotropic diffusion*, where the diffusion tensor is defined as

$$\mathbf{D}_{\mu}^{\text{iso}} = \text{diag}(1, 1, \mu^2). \tag{2.84}$$

the spatial plane and the diffusion in θ-direction *induce* smoothing in the other direction. This is the case since the Hörmander condition is also satisfied for $D_{\eta\eta} = 0$, see [39, Section 5.3].

The equation is "μ-isoptropic" since $\|\partial_\xi\|_\mu = \|\partial_\eta\|_\mu = \mu^2\|\partial_\theta\|_\mu = \mu$, so with respect to our μ-dependent definition of length of a vector cf. (2.69) we apply the same amount of diffusion in all directions. We use this convention for μ-isotropy because it ensures that the result of a β-isotropic diffusion process at time t can be identified one-to-one with the standard isotropic diffusion process on the image cf. (2.80) at the same value for time t on the image. This makes it more convenient to set parameter μ.

The μ-istropic diffusion equation can be written as

$$\partial_t W = \left(\left(\partial_\xi^2 + \partial_\eta^2\right) + \mu^2\partial_\theta^2\right) W = \left(\left(\partial_x^2 + \partial_y^2\right) + \mu^2\partial_\theta^2\right) W. \tag{2.85}$$

Since the operators ∂_x, ∂_y, and ∂_θ commute, this equation is the same as the diffusion equation in $\mathbb{R}^3$ except for the 2π-periodicity of the θ dimension.

2.9.1.2 Special Case: Anisotropic Diffusion Tangent to Exponential Curves

For anisotropic diffusion it is of interest to diffuse tangent to exponential curves with tangent vector $\boldsymbol{c} \in T_g(SE(2))$. In that case, the diffusion tensor is given by

$$\mathbf{D}_\mu(\boldsymbol{c}) = \frac{\mu^2}{\|\boldsymbol{c}\|_\mu^2}\, \boldsymbol{c} \cdot \boldsymbol{c}^{\mathrm{T}}, \tag{2.86}$$

where $\frac{\mu^2}{\|\boldsymbol{c}\|_\mu^2}$ is a matter of convention, ensuring that the diffusion process is independent of the length of $\boldsymbol{c}$, and that $\mathbf{D}_\mu(\boldsymbol{e}_\xi) + \mathbf{D}_\mu(\boldsymbol{e}_\eta) + \mathbf{D}_\mu(\boldsymbol{e}_\theta) = \mathbf{D}_\mu^{\mathrm{iso}}$

We can also combine μ-isotropic diffusion and anisotropic diffusion along exponential curves, where a parameter D_{a} controls the amount of isotropic resp. anisotropic diffusion, yielding for the diffusion tensor

$$\mathbf{D}_\mu(\boldsymbol{c}, D_{\mathrm{a}}) = (1 - D_{\mathrm{a}})\frac{\mu^2}{\|\boldsymbol{c}\|_\mu^2}\, \boldsymbol{c}\,\boldsymbol{c}^{\mathrm{T}} + D_{\mathrm{a}}\,\mathrm{diag}(1, 1, \mu^2), \tag{2.87}$$

so $D_{\mathrm{a}} = 0$ leads to completely anisotropic diffusion, and $D_{\mathrm{a}} = 1$ leads to μ-isotropic diffusion. Figure 2.9 shows examples of Green's functions for some different cases. In Chapter 6 we will use this type of diffusion in a nonlinear diffusion process.

2.9.1.3 Does Diffusion on $SE(2)$ Constitute a Scale Space?

Scale spaces are usually considered on scalar-valued images, i.e. functions on the translation group $\mathbb{R}^n$. The most widely used type of scale space on $\mathbb{R}^n$ is the Gaussian scale space [74, 152, 125], which is obtained by solving the diffusion equation

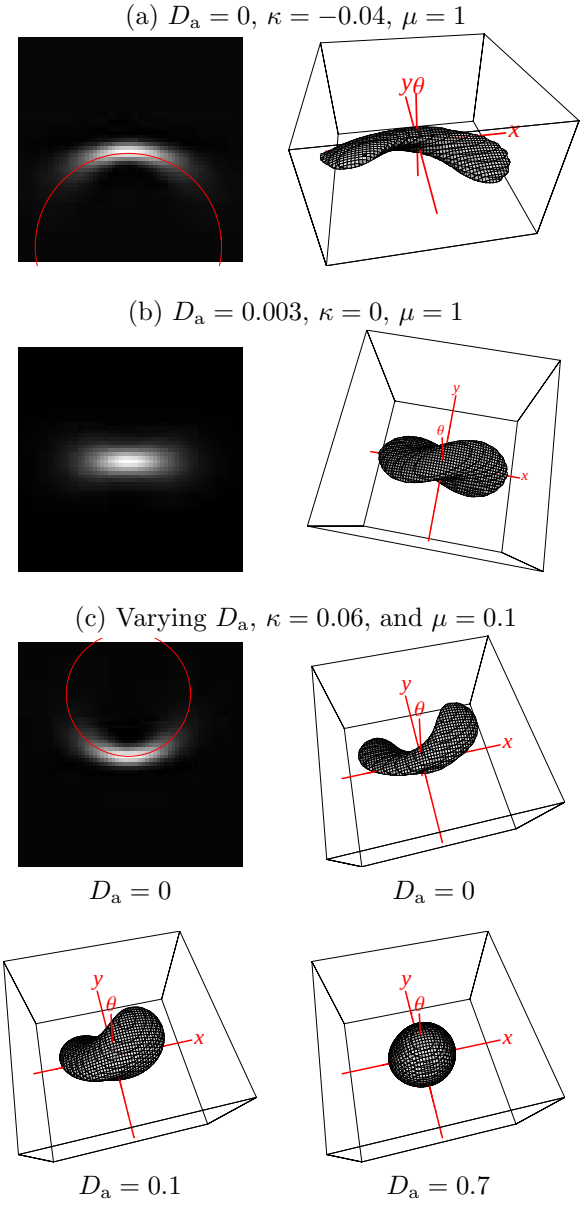


Figure 2.9: Illustrations of Green's functions of the $SE(2)$ diffusion equation (2.81) for different parameter values (obtained using an explicit iterative numerical scheme that will be introduced in Subsection 6.4.2) with in all examples time $t = 70$ and $d_H = 0$. (a) Shows the effect of nonzero κ. Left: Greens functions in the spatial plane (i.e. all orientations are summed) where the superimposed circular arc shows the expected curvature, Right: isosurface in 3D. (b) Shows the effect of a nonzero D_a. (c) Shows the effect of varying D_a. As D_a increases from 0 to 1, the resulting Green's function becomes more and more isotropic.

on the image. An operator $\mathbb{R}^+ \times \mathbb{L}_2(\mathbb{R}^n) \to \mathbb{L}_2(\mathbb{R}^n)$ generates a scale space if it satisfies a number of axioms. Following Duits [38], the necessary scale space axioms are: translation invariance, rotation invariance, bounded linear operator, scaling invariance, semigroup property, convergence, preservation of positivity, and average grey-value invariance. One could wonder whether our evolution equations satisfy a similar set of axioms on $SE(2)$. This question is posed purely out of curiosity, since for our purposes we do not need the diffusion equation to be a true scale space.

Most axioms of scale spaces on $\mathbb{R}^n$ directly map to $SE(2)$, where the equivalent of translation invariance is left-invariance. However, *rotation invariance* is not a desirable axiom on $SE(2)$, since in $SE(2)$ there is no inherent notion of isotropy, which would be necessary for a definition of rotation invariance. Therefore we drop this requirement, meaning that *all* linear diffusion equations cf. (2.81) might constitute scale spaces, in the sense that the diffusion tensor $\mathbf{D}$ parameterizes the class of diffusion scale spaces on $SE(2)$.

The most problematic axiom to map to $SE(2)$ is *scale invariance*. For this purpose a proper scaling operator should be defined on $SE(2)$, however, scaling the θ-dimension does not make sense. The only sensible scaling operator on orientation scores might be to scale the spatial dimensions only, such that scaling the orientation scores is the same as scaling the corresponding image, i.e.

$$(\Sigma_\lambda^{SE(2)} U)(x, y, \theta) = U(\frac{x}{\lambda}, \frac{y}{\lambda}, \theta). \tag{2.88}$$

The problem is that if we want the scaling to commute with diffusion on $SE(2)$ we now have to change the diffusion tensor, i.e. we have the relation

$$\Theta_t^{\mathbf{D}}(\Sigma_\lambda^{SE(2)}(U)) = \Sigma_\lambda^{SE(2)}(\Theta_{\lambda^2 t}^{\mathbf{\Lambda}(\lambda)\cdot\mathbf{D}\cdot\mathbf{\Lambda}(\lambda)}(U)), \tag{2.89}$$

where $\Theta_t^{\mathbf{D}}$ denotes the diffusion at time t with diffusion tensor $\mathbf{D}$ and where $\mathbf{\Lambda}(\lambda) = \mathrm{diag}(1, 1, \lambda)$. Notice that on the left-hand side and the right-hand side we have a *different* members of the class of possible diffusion scale spaces. For this reason, although most scale space axioms are satisfied, diffusion on $SE(2)$ does not constitute a true scale space on $SE(2)$.

Notice, however, that due to linearity of the orientation score transformation and due to left-invariance, linear diffusion on an orientation score U_f *does* constitute an $\mathbb{R}^2$-scale space on the corresponding image f, because the axiom of left-invariance ensures rotation invariance on the corresponding image.

2.9.2 Convection-Diffusion Equations on $SE(2)$

Besides diffusion equations, it is also of interest to study evolution equations with convection and diffusion, given by

$$\begin{cases} \partial_t W(g,t) = (-\boldsymbol{a}\cdot\nabla + \nabla\cdot\mathbf{D}\nabla)\, W(g;t), \quad \boldsymbol{x}\in\mathbb{R}^n, \quad t\geq 0, \\ W(g;0) = U_f(g), \end{cases} \tag{2.90}$$

where $\boldsymbol{a}$ is a vector describing the direction and speed of the convection part of the process.

In the case of convection and diffusion we are not only interested in the time process, but also in the resolvent process, given by

$$R_\lambda(\cdot) = \int_0^\infty W(\cdot,t) e^{-\lambda\, t}\mathrm{d}t, \tag{2.91}$$

where λ is a decay parameter. The solutions can be written as

$$\begin{aligned} W(\cdot,t) &= \exp(t\,A) U_f \\ R_\lambda &= \int_0^\infty \exp(t(A-\lambda)) U_f(\cdot)\mathrm{d}t = (A-\lambda)^{-1} U_f. \end{aligned} \tag{2.92}$$

In [35] exact expressions and practical approximations for the Green's function of this process are derived. The relevance of these resolvent processes will become apparent for the subsequent special case.

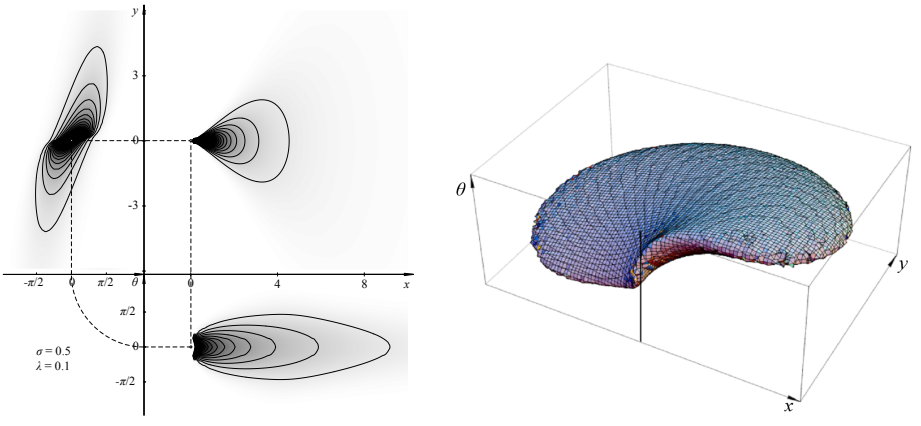
2.9.2.1 Special Case: Stochastic Completion Kernels

A special case is the so-called *stochastic completion kernel* which is the resolvent of the process above where (using the left-invariant basis)

$$\boldsymbol{a} = (1,0,0), \quad \mathbf{D} = \mathrm{diag}(0,0,\sigma^2). \tag{2.93}$$

This special case was first proposed by Mumford [103]. This model for lines and contours is based on a single assumption: the prior probability distribution of a curve can be modeled by a random walker in $SE(2)$. The equations for such a random walker are

$$\begin{aligned} \boldsymbol{x}(t) &= \int_{t_0}^{t} (\cos\theta(t'), \sin\theta(t')) + \boldsymbol{x}_0\, \mathrm{d}t' \\ \theta(t) &= \sigma^2 \epsilon_\theta(t) + \theta_0, \end{aligned} \tag{2.94}$$



(a) (b)

Figure 2.10: Illustrations of the completion kernel, taken from [2]. (a) The three marginals of the kernel. (b) 3D iso-surface of a stochastic completion kernel.

where t denotes time, $\epsilon_\theta(t)$ is a normally distributed stochastic variable with mean 0 and variance 1, σ is a parameter determining the standard deviation of the random walker, t_0 is the starting time and $(\boldsymbol{x}_0, \theta_0)$ is the starting position in $SE(2)$. From this equations one can obtain the probability distribution $p(\boldsymbol{x}, \theta; t|\boldsymbol{x}_0, \theta_0; t_0)$, which is the probability to find a random walker at a specific position and time given the initial conditions, yielding the following partial differential equation

$$\partial_t p(\boldsymbol{x}, \theta; t|\boldsymbol{x}_0, \theta_0; t_0) = (\sigma^2 \partial_\theta^2 - \partial_\xi - \lambda) p(\boldsymbol{x}, \theta; t|\boldsymbol{x}_0, \theta_0; t_0), \tag{2.95}$$

where $\lambda > 0$ is a "killing term", modeling the chance that a random walker ends, which can happen at any moment with a probability $1/\lambda$. A stochastic completion kernel is the Green's function of this differential equation and can be interpreted as the probability that a random walker starting at position $(\boldsymbol{x}_0, \theta_0) = (\mathbf{0}, 0)$ visits position $(\boldsymbol{x}, \theta)$ within the time-span $[0, \infty]$. This means one has to solve the above partial differential equation with a delta spike as initial condition, i.e. $p(\boldsymbol{x}, \theta) = \delta(\boldsymbol{x})\delta(\theta)$, and integrate over time

$$\begin{aligned} K(\boldsymbol{x}, \theta) &= \int_{t'=0}^{\infty} p(\boldsymbol{x}, \theta; t'|\mathbf{0}, 0; 0) \mathrm{d}t' \\ &= \int_{t'=0}^{\infty} \exp(t'(\sigma^2 \partial_\theta^2 - \partial_\xi - \lambda)) \delta(\boldsymbol{x})\delta(\theta) \mathrm{d}t' \\ &= -(\sigma^2 \partial_\theta^2 - \partial_\xi - \lambda)^{-1} \delta(\boldsymbol{x})\delta(\theta). \end{aligned} \tag{2.96}$$

Figure 2.10 gives an example of a completion kernel.

The idea of *stochastic completion field* by Williams and Jacobs [151] is to fill gaps in interrupted contours, by taking the product of a forward and backward process. This is accomplished by applying an $SE(2)$-convolution of a forward completion kernel K and a backward completion kernel $\tilde{K}(\boldsymbol{x}, \theta) = K(-\boldsymbol{x}, \theta + \pi)$,

$$C(\boldsymbol{x}, \theta) = (K *_{SE(2)} U)(\boldsymbol{x}, \theta) \cdot (\tilde{K} *_{SE(2)} U)(\boldsymbol{x}, \theta), \tag{2.97}$$

where U is the initial distribution and C the resulting stochastic completion field. U is interpreted as the local probability of having a line structure at position $\boldsymbol{x}$ with orientation θ. In most literature [151] [155], U is obtained by manually putting spikes in the orientation score, but also attempts were made to automate this [2] [27].

In Section 3.8 we will describe this approach for contour completion in more detail, including how to implement stochastic completion kernels in a steerable way.

A great deal of my work is just playing with equations and seeing what they give.

Paul Adrien Maurice Dirac (1902–1984)

3 Steerable $SE(2)$-Convolution

3.1 Introduction

In the previous chapter we introduced the basic principles of orientation scores. We pointed out the importance of left-invariance and consequently the importance of $SE(2)$-convolution for left-invariant *linear* operations. In this chapter we will consider the $SE(2)$-convolution in more detail. First we will treat its mathematical properties. Subsequently, we will describe how to implement the $SE(2)$-convolution straightforwardly. The fact that we need to rotate the $SE(2)$ convolution kernel is problematic, because it is often necessary to interpolate the kernel. To solve this problem, we show that we can generalize the well-known concept of rotationally *steerable filters* on images [56] to steerable $SE(2)$-convolution kernels on orientation scores. The resulting method for steerable $SE(2)$-convolution is more efficient in many cases. The novelty of this method is that it applies rotationally steerable filters on orientation scores instead of on 2D or 3D images.

Since rotations constitute a Lie group, Lie group theory can be used to analyze steerability, which has already been exploited in the context of steerable filters on images by Michaelis et al. [100] and Teo and Hel-Or [132, 70]. We will show that there is a clear relation between the steerable $SE(2)$-convolution that we will derive and the Fourier transform on $SE(2)$ [22]. In fact, our approach for steerable $SE(2)$-convolution is similar to using the convolution theorem on $SE(2)$ cf. [22], however our approach prevents unnecessary computations as we do not need to calculate the Fourier transform on $SE(2)$ explicitly. This is especially beneficial as the polar interpolation step needed to calculate the $SE(2)$-Fourier transform is omitted.

Next, we show that an orientation score that is sampled in the orientation dimension imposes a maximum bandwidth of the steerable $SE(2)$-kernel. This means that all $SE(2)$-kernels should in fact be steerable kernels. Subsequently, we will discuss some special cases appearing in practice under which the steerable $SE(2)$-convolution can be simplified.

Finally, we will discuss how $SE(2)$-convolutions relate to existing image processing techniques. We will apply steerable $SE(2)$-convolutions to calculate stochastic completion fields. We will also show how channel smoothing can be regarded as an $SE(2)$-convolution.

3.2 Definition of Transforms

For the sake of clarity, we will define the elementary transforms that are used throughout this chapter.

3.2.1 Fourier Transforms and Series

The *spatial Fourier* transform on $f : \mathbb{R}^2 \to \mathbb{C}$ was already defined in (2.10) as

$$\mathcal{F}_{\mathbb{R}^2}[f](\boldsymbol{\omega}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} f(\boldsymbol{x}) e^{-i\boldsymbol{\omega}\cdot\boldsymbol{x}} \mathrm{d}\boldsymbol{x}, \tag{3.1}$$

where $\boldsymbol{x} = (x, y)$ and $\boldsymbol{\omega} = (\omega_x, \omega_y)$. The inverse is given by

$$\mathcal{F}^{-1}_{\mathbb{R}^2}[\hat{f}](\boldsymbol{x}) = \frac{1}{2\pi} \int_{\mathbb{R}^2} f(\boldsymbol{x}) e^{i\boldsymbol{\omega}\cdot\boldsymbol{x}} \mathrm{d}\boldsymbol{\omega}. \tag{3.2}$$

Convolution on $\mathbb{R}^2$ is defined as

$$(f *_{\mathbb{R}^2} g)(\boldsymbol{x}) = \int_{\mathbb{R}^2} f(\boldsymbol{x} - \boldsymbol{x}') g(\boldsymbol{x}') \mathrm{d}\boldsymbol{x}', \tag{3.3}$$

and the convolution theorem is given by

$$\mathcal{F}_{\mathbb{R}^2}[f *_{\mathbb{R}^2} g](\boldsymbol{\omega}) = 2\pi\, \mathcal{F}_{\mathbb{R}^2}[f](\boldsymbol{\omega}) \cdot \mathcal{F}_{\mathbb{R}^2}[g](\boldsymbol{\omega}), \tag{3.4}$$

where the occurence of factor 2π is a consequence of using the unitary convention for the Fourier transform.

The *angular Fourier* series on a function $f : \mathbb{T} \to \mathbb{C}$ is defined by

$$\mathcal{F}_{\mathbb{T}}[f]_m = \hat{f}_m = \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} f(\theta) e^{-im\theta} \mathrm{d}\theta, \tag{3.5}$$

and the inverse is

$$\mathcal{F}^{-1}_{\mathbb{T}}[\hat{f}](\theta) = f(\theta) = \frac{1}{\sqrt{2\pi}} \sum_{m\in\mathbb{Z}} \hat{f}_m e^{im\theta}. \tag{3.6}$$

Convolution on $\mathbb{T}$ is defined as

$$(f *_{\mathbb{T}} g)(\theta) = \int_0^{2\pi} f(\theta - \theta') g(\theta') \mathrm{d}\theta', \tag{3.7}$$

and the convolution theorem is given by

$$\mathcal{F}_{\mathbb{T}}[f *_{\mathbb{T}} g]_m = \sqrt{2\pi}\, \mathcal{F}_{\mathbb{T}}[f]_m \cdot \mathcal{F}_{\mathbb{T}}[g]_m. \tag{3.8}$$

The 1D discrete Fourier transform (DFT) of sampled signal $\boldsymbol{a} = (a_0, a_1, \ldots, a_{N-1})$ is defined by

$$\mathrm{DFT}[\boldsymbol{a}]_p = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} a_k e^{-i\frac{2\pi}{N}kp}, \tag{3.9}$$

and the inverse is given by

$$\mathrm{DFT}^{-1}[\hat{\boldsymbol{a}}]_k = \frac{1}{\sqrt{N}} \sum_{p=0}^{N-1} \hat{a}_p e^{i\frac{2\pi}{N}kp}. \tag{3.10}$$

Since DFT is separable, this trivially extends to any-dimensional data by concatenation of 1D DFT operations in all direction. For example, a DFT of a 2D sampled $N \times N$ image $\mathbf{f} = [f_{p,q}]$, $p \in \{0, 1, \ldots, N\}$ and $q \in \{0, 1, \ldots, N\}$ is given by

$$\mathrm{DFT}_{2\mathrm{D}}[\mathbf{f}]_{k,l} = \mathrm{DFT}[q \mapsto \mathrm{DFT}[p \mapsto f_{p,q}]_m]_n, \tag{3.11}$$

and

$$\mathrm{DFT}_{2\mathrm{D}}^{-1}[\hat{\mathbf{f}}]_{k,l} = \mathrm{DFT}^{-1}[n \mapsto \mathrm{DFT}^{-1}[m \mapsto \hat{f}_{m,n}]_p]_q. \tag{3.12}$$

3.2.2 The Hankel Transform

The Hankel transform is defined as

$$\mathcal{H}_m[f](\rho) = i^m \int_0^\infty f_m(r) J_m(\rho\, r) r \,\mathrm{d}r, \tag{3.13}$$

where J_m is the mth order Bessel function of the first kind. The inverse Hankel transform is given by

$$\mathcal{H}_m^{-1}[\hat{f}](r) = (-i)^m \int_0^\infty \hat{f}_m(\rho) J_m(\rho\, r) \mathrm{d}\rho. \tag{3.14}$$

Consider a function $F_m(\boldsymbol{x}) = f_m(r)e^{im\phi}$, with $\boldsymbol{x} = (r\cos\phi, r\sin\phi)$. The following relation holds

$$\mathcal{F}_{\mathbb{R}^2}[F_m](\omega_r \cos\varphi, \omega_r \sin\varphi) = 2\pi\, \mathcal{H}_m[f_m](\omega_r) e^{im\varphi}. \tag{3.15}$$

Consequently, if $G_n(\boldsymbol{x}) = g_n(r)e^{in\phi}$, then

$$(F_m *_{\mathbb{R}^2} G_n)(\boldsymbol{x}) = (2\pi)^3\, \mathcal{H}_{m+n}^{-1}[\mathcal{H}_m[f_m]\mathcal{H}_n[g_n]](r) e^{i(m+n)\phi}. \tag{3.16}$$

3.3 The $SE(2)$-Convolution

Remember from Subsection 2.7.3 that the $SE(2)$-convolution is defined by

$$(\Psi *_{SE(2)} U)(\boldsymbol{x}, \theta) = \int_{\mathbb{R}^2} \int_0^{2\pi} \Psi(\mathbf{R}_{\theta'}^{-1}(\boldsymbol{x} - \boldsymbol{x}'), \theta - \theta') U(\boldsymbol{x}', \theta') \,\mathrm{d}\theta' \,\mathrm{d}\boldsymbol{x}'. \tag{3.17}$$

where $U \in \mathbb{L}_2(SE(2))$ is an orientation score of an image, and $\Psi \in \mathbb{L}_2(SE(2))$ is the $SE(2)$-convolution kernel. In Figure 2.4 on page 30 we schematically showed how an $SE(2)$-convolution is performed.

3.3.1 Properties of the $SE(2)$-Convolution

Most properties of "normal" $\mathbb{R}^n$-convolution also apply to $SE(2)$-convolution. The most important properties are summarized below

- *Associativity*:

$$(\Psi_1 *_{SE(2)} \Psi_2) *_{SE(2)} U = \Psi_1 *_{SE(2)} (\Psi_2 *_{SE(2)} U). \tag{3.18}$$

  This also holds for scalar multiplication:

$$a(\Psi *_{SE(2)} U) = ((a\Psi) *_{SE(2)} U) = (\Psi *_{SE(2)} (aU)), \tag{3.19}$$

  for all $a \in \mathbb{C}$;

- *Distributivity*:

$$(\Psi_1 + \Psi_2) *_{SE(2)} U = \Psi_1 *_{SE(2)} U + \Psi_2 *_{SE(2)} U, \tag{3.20}$$

  and

$$\Psi *_{SE(2)} (U_1 + U_2) = \Psi *_{SE(2)} U_1 + \Psi *_{SE(2)} U_2; \tag{3.21}$$

- *Identity*: $\delta *_{SE(2)} U = U *_{SE(2)} \delta = U$ where $\delta(x, y, \theta) = \delta(x)\delta(y)\delta(\theta)$;

- *Invariance*: an $SE(2)$-convolution is left-invariant (see Section 2.7) on its right-input,

$$\mathcal{L}_g(\Psi *_{SE(2)} U) = \Psi *_{SE(2)} (\mathcal{L}_g U), \ \forall g \in SE(2), \tag{3.22}$$

  and right-invariant on its left-input,

$$\mathcal{Q}_g(\Psi *_{SE(2)} U) = (\mathcal{Q}_g \Psi) *_{SE(2)} U, \ \forall g \in SE(2). \tag{3.23}$$

  This is the same as the translation-invariance property of normal convolution. Furthermore, we have the following properties

$$(\mathcal{L}_g \Psi) *_{SE(2)} U = \Psi *_{SE(2)} (\mathcal{Q}_{g^{-1}} U), \ \forall g \in SE(2), \tag{3.24}$$

  and

$$\Psi *_{SE(2)} (\mathcal{Q}_g U) = (\mathcal{L}_{g^{-1}} \Psi) *_{SE(2)} U, \ \forall g \in SE(2). \tag{3.25}$$

- *Differentiation rule*: $\mathcal{D}(\Psi *_{SE(2)} U) = ((\mathcal{D}\Psi) *_{SE(2)} U)$ where $\mathcal{D}$ denotes a *left-invariant* derivative operator. This property can be easily derived by rewriting (2.52) as follows (where $\mathcal{L}_g$ is defined in (2.33))

$$\begin{aligned} \mathcal{D} \int_{SE(2)} (\mathcal{L}_h \circ \Psi)(g) U(h) dh &= \int_{SE(2)} (\mathcal{D} \circ \mathcal{L}_h \circ \Psi)(g) U(h) dh \\ &\overset{\mathcal{D} \text{ left-inv.}}{=} \int_{SE(2)} (\mathcal{L}_h \circ (\mathcal{D} \circ \Psi))(g) U(h) dh. \end{aligned} \tag{3.26}$$

  Furthermore, $\mathcal{D}(\Psi *_{SE(2)} U) = (\Psi *_{SE(2)} (\mathcal{D}U))$ iff $\mathcal{D}$ is *right-invariant*, which follows by (where $\mathcal{Q}_h$ is defined in (2.34))

$$\begin{aligned} &\mathcal{D} \int_{SE(2)} \Psi(h^{-1}g) U(h) dh \overset{\text{subst. } h \leftarrow gh}{=} \mathcal{D} \int_{SE(2)} \Psi(h^{-1}) (\mathcal{Q}_h \circ U)(g) dh \\ &= \int_{SE(2)} \Psi(h^{-1}) (\mathcal{D} \circ \mathcal{Q}_h \circ U)(g) dh \\ &\overset{\mathcal{D} \text{ right-inv.}}{=} \int_{SE(2)} \Psi(h^{-1}) (\mathcal{Q}_h \circ (\mathcal{D} \circ U))(g) dh \\ &\overset{\text{subst. } h \leftarrow g^{-1}h}{=} \int_{SE(2)} \Psi(h^{-1}g) (\mathcal{D}U)(h) dh. \end{aligned} \tag{3.27}$$

  Note that these two properties do not only hold for differential operators: all linear left-invariant operators commute with the left-input of the $SE(2)$-convolution while all linear right-invariant operators commute with the right-input of the $SE(2)$-convolution.

- *Convolution theorem*: this property will be addressed in Section 3.6.

In contrast to an $\mathbb{R}^n$-convolution, the $SE(2)$-convolution is not commutative: $\Psi *_{SE(2)} U \neq U *_{SE(2)} \Psi$. However, the following relation holds

$$(\Psi *_{SE(2)} U)(g) = (\check{U} *_{SE(2)} \check{\Psi})(g^{-1}), \quad \forall g \in SE(2), \tag{3.28}$$

with $\check{U}(g) = U(g^{-1})$ and $\check{\Psi}(g) = \Psi(g^{-1})$.

3.3.2 $SE(2)$-Convolution versus $SE(2)$-Correlation

Analogously to the $\mathbb{R}^2$-correlation cf. (2.7), the $SE(2)$-*correlation* is defined by

$$(\check{\Psi} \star_{SE(2)} U)(g) = \int_{SE(2)} \check{\Psi}(g^{-1}h) U(h) dh, \tag{3.29}$$

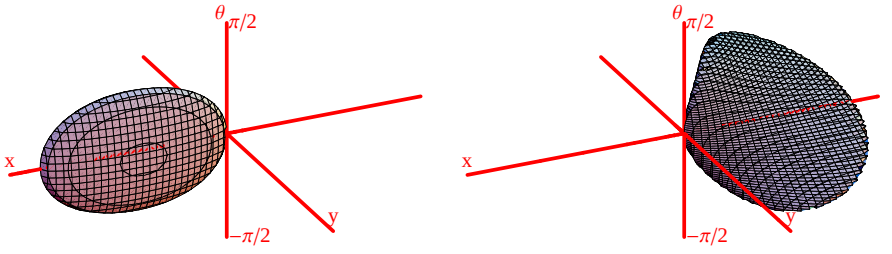


Figure 3.1: Example showing the effect of "mirroring" an $SE(2)$-kernel. Left: an arbitrary $SE(2)$-kernel Ψ. Right: $\breve{\Psi}$ according to $\breve{\Psi}(\boldsymbol{x},\theta) = \Psi(-\mathbf{R}_\theta^{-1}\boldsymbol{x},-\theta)$, this is the mirrored version of Ψ with an additional twist over the orientation dimension. An $SE(2)$-convolution with the kernel on the left is the same as an $SE(2)$-correlation with the kernel on the right and vice versa.

where $\breve{\Psi}(g) = \Psi(g^{-1})$ so $\Psi *_{SE(2)} U = \breve{\Psi} \star_{SE(2)} U$. More explicitly this yields

$$(\breve{\Psi} \star_{SE(2)} U)(\boldsymbol{x},\theta) = \int_{\mathbb{R}^2} \int_0^{2\pi} \breve{\Psi}(\mathbf{R}_\theta^{-1}(\boldsymbol{x}' - \boldsymbol{x}), \theta' - \theta) U(\boldsymbol{x}', \theta') \, \mathrm{d}\theta' \, \mathrm{d}\boldsymbol{x}', \tag{3.30}$$

where $\breve{\Psi}(\boldsymbol{x},\theta) = \Psi(-R_\theta^{-1}\boldsymbol{x},-\theta)$. We see that because of the structure of $SE(2)$, transforming an $SE(2)$-convolution kernel into an $SE(2)$-correlation kernel $\breve{\Psi}$ is not accomplished by only mirroring the x, y, and θ-axes, but there is also a rotation involved. This can be graphically visualized as "twisting the kernel", see Figure 3.1. Consequently, the kernels Ψ and $\breve{\Psi}$ can have quite different appearances.

Further on, we will always use $SE(2)$-convolution instead of $SE(2)$-correlation. The most important reason is that $SE(2)$-correlation does not fulfill the properties of associativity, identity, and the differentiation rule. We will not explicitly give the derivations of the (steerable) $SE(2)$-correlation, as these are straightforward to obtain from the equivalent derivations of the (steerable) $SE(2)$-convolution.

3.3.3 Implementation of $SE(2)$-Convolution

In order to discretize orientation scores we take equidistant samples in the $SE(2)$ group using the standard (x, y, θ) coordinates. In the spatial dimensions, the sampling distance is set equal to the sampling distance of the corresponding image. We assume for simplicity that we deal with a square image with $N_s \times N_s$ pixels. In the orientation dimension, the number of samples to take is a parameter denoted by N_o, so the sampling distance is given by $s_\theta = 2\pi/N_o$ for a 2π-periodic orien-

Symbol	*Meaning*
N_s	Spatial dimensions of the orientation score or image (assuming equal x and y dimensions)
N_o	Number of sampled orientations of the orientation score U
K_s	Spatial dimensions of $SE(2)$-kernel Ψ
K_o	Number of sampled orientations of $SE(2)$-kernel Ψ
C_s	Number of spatial-angular frequency components of $SE(2)$-kernel Ψ
C_o	Number of orientation-angular frequency components of $SE(2)$-kernel Ψ

Table 3.1: Symbols used in the complexity estimates for different $SE(2)$-convolution implementations in this chapter. The following relations between the quantities are assumed: $C_s \leq N_o$ and $C_o \leq N_o$, which follows from the sampling theorem on $SE(2)$-kernels in Subsection 3.5.3.2; $K_s < N_s$ and $K_o < N_o$, stating that an $SE(2)$-kernel should not be larger than the orientation score; and $N_o \leq N_s$, i.e. more samples in the spatial dimensions than in the orientation dimension.

$SE(2)$-convolution implementation	*Section*	*Complexity*
Non-steerable implementation direct	3.3.3	$\mathcal{O}\left(N_o K_o N_s^2 K_s^2\right)$
Non-steerable implementation FFT	3.3.3	$\mathcal{O}\left(N_o N_s^2 (\log N_s + K_o)\right)$
Steerable implementation direct	3.5.2	$\mathcal{O}\left(C_o C_s N_s^2 K_s^2\right)$
Steerable implementation FFT	3.5.2	$\mathcal{O}\left(N_s^2 (N_o \log N_s + C_o C_s)\right)$
$SE(2)$-Fourier, spline interpolation [a]	3.6.2	$\mathcal{O}\left(N_s^2 (N_o \log N_s + {N_o}^{\zeta})\right)$
$SE(2)$-Fourier, Fourier interpolation [a]	3.6.2	$\mathcal{O}\left(N_s^2 \left(N_o (\log N_s)^2 + {N_o}^{\zeta}\right)\right)$
Special cases		
Non-steerable lifted orientation score direct[b]	3.7.3	$\mathcal{O}\left(N_o N_s^2 K_s^2\right)$
Steerable lifted orientation score direct	3.7.3	$\mathcal{O}\left(C_o C_s N_s^2 K_s^2\right)$
Steerable lifted orientation score FFT	3.7.3	$\mathcal{O}\left(N_s^2 (N_o \log N_s + C_o C_s)\right)$
Non-steerable separable $SE(2)$-kernel direct[c]	3.7.4	$\mathcal{O}\left(N_o N_s^2 K_s^2\right)$
Non-steerable separable $SE(2)$-kernel FFT[c]	3.7.4	$\mathcal{O}\left(N_o N_s^2 \log N_s\right)$
Steerable tensorial convolution direct[d]	3.7.4	$\mathcal{O}\left(N_s^2 K_s^2\right)$
Steerable tensorial convolution FFT[d]	3.7.4	$\mathcal{O}\left(N_s^2 \log N_s\right)$
Non-steerable tensor voting direct[b]	4.3.1	$\mathcal{O}\left(N_s^2 K_s^2\right)$
Steerable tensor voting direct	4.3.1	$\mathcal{O}\left(C_s N_s^2 K_s^2\right)$
Steerable tensor voting FFT	4.3.1	$\mathcal{O}\left(C_s N_s^2 \log N_s\right)$

[a] With $2 \leq \zeta \leq 3$, depending on the complexity of matrix multiplication.
[b] An FFT implementation is not possible in this case.
[c] A steerable implementation is senseless in this case, since the complexity multiplies with a factor C_s.
[d] A non-steerable implementation does not make sense in this case.

Table 3.2: Overview of the algorithmic complexities of the different algorithms for the $SE(2)$-convolution discussed in this chapter and the next chapter. The symbols used in the algorithmic complexities are specified in Table 3.1.

tation score and $s_\theta = \pi/N_o$ for a π-periodic orientation score[1]. Table 3.1 gives an overview of the symbols that we use to specify the sizes of orientation scores and $SE(2)$-kernels, where we assume that the kernels are also spatially square. In Table 3.2 an overview is given of the complexities of the different algorithms that will be discussed in this chapter and.

The most straightforward implementation of $SE(2)$-convolution is obtained by replacing the integrals in (3.17) by finite sums. We index the samples of the $SE(2)$-kernel by $\Psi(x, y, l)$ with $x, y \in \{-K_s^{\mathrm{H}}, \dots K_s^{\mathrm{H}}\}$, $K_s^{\mathrm{H}} = \lfloor K_s/2 \rfloor$, and $l \in \{-K_o^{\mathrm{H}}, \dots K_o^{\mathrm{H}}\}$, $K_o^{\mathrm{H}} = \lfloor K_o/2 \rfloor$, assuming for simplicity that the sampled Ψ has odd dimensions and the central value $\Psi(0, 0, 0)$ is represented by the middle sample. The samples of the orientation score are indexed by $U(x, y, l)$ with $x, y \in \{0, , 1, \dots N_s - 1\}$ and $l \in \{0, 1, \dots N_o - 1\}$. This yields

$$\begin{aligned} &R(x, y, l) \\ &= \sum_{l'=l-K_o^{\mathrm{H}}}^{l+K_o^{\mathrm{H}}} \sum_{x'=x-K_s^{\mathrm{H}}}^{x+K_s^{\mathrm{H}}} \sum_{y'=y-K_s^{\mathrm{H}}}^{y+K_s^{\mathrm{H}}} \mathcal{R}_{l' \cdot s_\theta}[\Psi](x - x', y - y', l - l') U_{\mathrm{ext}}(x', y', l'), \end{aligned} \tag{3.31}$$

where $\mathcal{R}_{l' \cdot s_\theta}[\Psi]$ denotes a rotated $SE(2)$-kernel and U_{ext} denotes the sampled orientation score U with an *extended* domain, i.e. it is defined on the entire domain $\mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$, taking into account the appropriate boundary conditions. For the orientation dimension the only appropriate boundary condition is periodic. For the spatial dimension, however, it is less obvious what boundary condition to choose, since all boundary conditions will create undesired discontinuities giving rise to boundary artefacts. Usually, the zero-padding boundary condition is chosen. The complexity of the implementation according to (3.31) is given by $\mathcal{O}\left(N_o K_o N_s^2 K_s^2\right)$. Note that a direct implementation of an $\mathbb{R}^3$-convolution has the same complexity.

The rotated kernels $\mathcal{R}_{l \cdot s_\theta}[\Psi]$ can be obtained in different ways. If one has an analytic formula for Ψ, it is most accurate to sample it for the N_o required rotations $l \cdot s_\theta$. The drawback is that this takes a large amount of memory. Alternatively, one can only sample Ψ for a single orientation, and rotate the kernel using interpolation. In our implementation we use B-spline interpolation proposed by Unser et al. [136]. This is an efficient interpolation algorithm consisting of a prefiltering step with a separable infinite impulse response filter to determine the B-spline coefficients, followed by inner product calculations with shifted B-splines to obtain interpolated values. Experiments have shown that using 2nd order B-spline interpolation leads to satisfactory interpolation results, while first order B-spline interpolation, which amounts to linear nearest neighbor interpolation, results in visible additional blurring of the kernel.

Alternatively, it is possible to make the algorithm more efficient using the 2D fast

[1] Note that for a π-periodic $SE(2)$-convolution the convolution kernel needs to fulfill $\Psi(\boldsymbol{x}, \theta) = \Psi(-\boldsymbol{x}, \theta)$ and $\Psi(\boldsymbol{x}, \theta) = \Psi(\boldsymbol{x}, \theta + \pi)$ for all $(\boldsymbol{x}, \theta) \in SE(2)$.

Fourier transform (FFT) algorithm, yielding

$$(\Psi *_{SE(2)} U)(x, y, l) = \sum_{l'=l-K_o^{\mathrm{H}}}^{l+K_o^{\mathrm{H}}} \mathrm{DFT}^{-1}[\mathrm{DFT}[\mathrm{PAD}_{N_s^2}[\mathcal{R}_{l' \cdot s_\theta}[\Psi(\cdot, \cdot, l - l')]]] \mathrm{DFT}[U(\cdot, \cdot, l')]](x, y), \quad (3.32)$$

where DFT denotes the 2D DFT cf. (3.11), which is implemented using FFT, and $\mathrm{PAD}_{N_s^2}$ denotes the process of padding zero-valued samples to the kernel such that it has the same spatial dimensions as the orientation score. The steps of the algorithm are:

1. 2D FFT of $U(\cdot, \theta)$ for each sampled θ: complexity $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$.
2. For each summand in (3.32) (K_o terms) and each sampled orientation (N_o terms) apply a multiplication for each spatial frequency component (N_s^2 terms): complexity $\mathcal{O}\left(K_o N_o N_s^2\right)$
3. 3D inverse FFT on the result: complexity $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$

The total complexity is $\mathcal{O}\left(N_o N_s^2 (\log N_s + K_o)\right)$. Note that this is a higher complexity than the complexity of a 3D Fourier transform via the Fourier domain, because the rotation of the kernel makes it impossible to take advantage of the FFT algorithm in the orientation dimension.

3.4 Steerable Filters

The $SE(2)$-convolution involves translations and rotations of the $SE(2)$-kernel. In practice, translation is easy to accomplish on a sampled grid, but rotation is more cumbersome. For anisotropic filtering of images a well-known solution to the problem of rotation is to use *steerable* filters [56]. In this section we will explain the concept of *steerability*, which we will apply later on to construct steerable $SE(2)$-convolutions.

3.4.1 Steerability

Suppose we want to filter an image f with an anisotropic filter h rotated over many different angles, i.e. we want to calculate $f *_{\mathbb{R}^2} \mathcal{R}_\alpha[h]$ for many values of α. The straightforward approach is to use the relation $\mathcal{R}_\alpha[h](\boldsymbol{x}) = h(\mathbf{R}_\alpha^{-1})$ to sample the rotated kernel for all requested rotations and apply convolutions. If we do not have an analytic expression for h, the rotated kernels then need to be obtained

by using some interpolation scheme. However, the calculation of $f *_{\mathbb{R}^2} \mathcal{R}_\alpha[h]$ for many values of α can be made more efficient and simpler if h is a steerable filter, which means that there exists a decomposition of h such that

$$\mathcal{R}_\alpha[h](\boldsymbol{x}) = \sum_{j=1}^{M} k_j(\alpha) h_j(\boldsymbol{x}), \;\; M < \infty, \tag{3.33}$$

where k_j are called the *steering functions* and h_j are the basis functions. Now we can write the convolution as

$$f *_{\mathbb{R}^2} \left(\sum_{j=1}^{M} k_j(\alpha) h_j \right) = \sum_{j=1}^{M} k_j(\alpha) \left(f *_{\mathbb{R}^2} h_j \right), \tag{3.34}$$

where the right-hand side is obtained by interchanging the order of the sum and the convolution operator. We observe that if we first convolve the image with all functions h_j, we can calculate the filter response in any orientation, simply by a linear combination with weighting coefficients $k_j(\alpha)$. In this way, it is not necessary anymore to convolve the image with the filter kernel in all required orientations, which can save a lot of computations if M is sufficiently small.

The applicability of this principle is wider than rotations on $\mathbb{R}^2$ only. In all cases where an efficient, possibly approximate, decomposition cf. (3.33) exist, one can take advantage of steerability for linear convolution operations on other domains than $\mathbb{R}^2$ as well as on other operations than rotations.

3.4.2 Irreducible Representations

Steerability is related to irreducible representations on groups. In this subsection we will explain the notion of irreducible representations and establish the link to steerability. In our explanation, we will use matrices rather than of operators for simplicity.

Suppose $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$ is a vector in space V, in which a representation $\mathcal{A}$ of group G exists, denoted by $\mathcal{A}_g[\boldsymbol{v}] = \mathbf{A}(g) \cdot \boldsymbol{v}$ where $g \in G$ and $\mathbf{A}$ is the matrix corresponding to the representation $\mathcal{A}$.

Now we try to find a transformation matrix $\mathbf{D}$ and construct a matrix $\tilde{\mathbf{A}}$ by

$$\tilde{\mathbf{A}}(g) = \mathbf{D}\, \mathbf{A}(g)\, \mathbf{D}^{-1}, \tag{3.35}$$

such that $\tilde{\mathbf{A}}(g)$ is block-diagonalized *as much as possible* for all values of g, i.e.

$\tilde{\mathbf{A}}(g)$ will have the following form

$$\tilde{\mathbf{A}}(g) = \begin{pmatrix} \boldsymbol{\chi}_1(g) & 0 & \cdots & 0 \\ 0 & \boldsymbol{\chi}_2(g) & \ddots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{\chi}_M(g) \end{pmatrix} = \bigoplus_{i=1}^{M} \boldsymbol{\chi}_i(g), \quad M \leq N. \tag{3.36}$$

The submatrices $\boldsymbol{\chi}_i(g)$ along the diagonal represent *irreducible representations* of G, which are submatrices that cannot be diagonalized any further. Vector $\boldsymbol{v}$ can now be transformed as $\tilde{\boldsymbol{v}} = \mathbf{D}\boldsymbol{v}$ and the group transformation is performed by $\mathcal{A}_g[\tilde{\boldsymbol{v}}] = \tilde{\mathbf{A}}(g) \cdot \tilde{\boldsymbol{v}}$. In this way the original $N \times N$ matrix multiplication simplifies to M matrix multiplications with the block-diagonal matrices $\boldsymbol{\chi}_i(g)$, which leads to a reduction in computational cost.

An important result from group theory is that for any *commutative* group, all irreducible representations are one-dimensional, so in that case $\boldsymbol{\chi}_i(\theta)$ are all "1×1 submatrices". For the 2D rotation group $SO(2)$, all 1D irreducible representations have the form

$$\chi_m(\alpha) = e^{-im\alpha} \text{ with } m \in \mathbb{Z}, \tag{3.37}$$

where it depends on the rotational properties of the object which values of m occur. For $SO(2)$ the angular Fourier series cf. (3.5) plays the same role as the transformation matrix $\mathbf{D}$ in the explanation above. The angular Fourier coefficients $\mathcal{F}_{\mathbb{T}}[\mathcal{R}_\alpha[f]]_m$ represent the function $f : \mathbb{T} \to \mathbb{C}$ in such a way that rotation is achieved by multiplication with the 1D irreducible representations of $SO(2)$, i.e.

$$\begin{aligned} \mathcal{F}_{\mathbb{T}}[\mathcal{R}_\alpha[f]]_m &= \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} f(\theta - \alpha) e^{-im\theta} \mathrm{d}\theta \quad \text{(subst. } \theta = \theta' + \alpha) \\ &= \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} f(\theta') e^{-im(\theta'+\alpha)} \mathrm{d}\theta' = e^{-im\alpha} \mathcal{F}_{\mathbb{T}}[f]_m = \chi_m(\alpha) \mathcal{F}_{\mathbb{T}}[f]_m. \end{aligned} \tag{3.38}$$

An object (e.g. a filter) is *steerable* if a *finite* amount of irreducible representations are needed to rotate the object. Steerable filters can therefore be constructed by using the irreducible representations of the group as the steering functions cf. (3.33). For example, a kernel $h : \mathbb{R}^2 \to \mathbb{R}$ can be made steerable by writing it in polar coordinates $\breve{h}(r, \phi) = h(r \cos\phi, r \sin\phi)$ and decomposing as follows

$$\mathcal{R}_\alpha[h](\boldsymbol{x}) = \sum_{m=-M}^{M} \underbrace{e^{-im\alpha}}_{k_m(\alpha)} \underbrace{\mathcal{F}_{\mathbb{T}}[\breve{h}(r, \cdot)]_m e^{im\phi}}_{h_m(\boldsymbol{x})}. \tag{3.39}$$

If h is angularly bandlimited, so $M < \infty$ then h is a steerable filter.

3.5 Orientation Scores and Steerability

In this section, we will first derive how the $SE(2)$-convolution can be made steerable. Then, we will discuss implementational details and consider the computational complexity. Subsequently, we will establish the relation between steerable $SE(2)$-convolution and the Fourier transform on $SE(2)$. Finally, we will discuss the consequences of sampling in the orientation dimension on the choices of orientation score construction kernel ψ and $SE(2)$kernel Ψ.

In this section we will write functions $SE(2) \to \mathbb{C}$, such as the $SE(2)$-kernel Ψ, in both spatially Cartesian coordinates $\Psi(\boldsymbol{x}, \theta) = \Psi(x, y, \theta)$ and spatially polar coordinates $\Psi(r, \phi, \theta)$ where $\boldsymbol{x} = (r\cos\phi, r\sin\phi)$. In the notation, the coordinate system in use is disambiguated by the use of the symbols (x, y) and (r, ϕ) respectively.

3.5.1 Steerable $SE(2)$-Convolution

In polar coordinates, the rotation of an $SE(2)$-kernel Ψ is given by $\mathcal{R}_\alpha[\Psi](r, \phi, \theta) = \Psi(r, \phi - \alpha, \theta - \alpha)$. Note that due to the group structure the rotations of ϕ and θ are *coupled*, meaning that it is senseless to consider cases where two separate rotations are applied in the ϕ-dimension and the θ-dimension.

To make an $SE(2)$-kernel steerable, we apply an angular Fourier series decomposition in both ϕ and θ. We define the *full-angular Fourier decomposition* $\mathcal{S} : (SE(2) \to \mathbb{C}) \to (\mathbb{Z} \times \mathbb{Z} \times \mathbb{R}^+ \to \mathbb{C})$:

$$\mathcal{S}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}(r) = \tfrac{1}{2\pi} \int_0^{2\pi} \int_0^{2\pi} \Psi(r, \phi, \theta) e^{-i((m_\mathrm{s} - m_\mathrm{o})\phi + m_\mathrm{o}\theta)} \mathrm{d}\theta \, \mathrm{d}\phi, \tag{3.40}$$

$$\mathcal{S}^{-1}[\hat{\Psi}](r, \phi, \theta) = \tfrac{1}{2\pi} \sum_{m_\mathrm{s} \in \mathbb{Z}} \sum_{m_\mathrm{o} \in \mathbb{Z}} \hat{\Psi}_{m_\mathrm{o}, m_\mathrm{s}}(r) e^{i((m_\mathrm{s} - m_\mathrm{o})\phi + m_\mathrm{o}\theta)}, \tag{3.41}$$

where m_o refers to the frequency components in the orientation dimension, and $m_\mathrm{s} - m_\mathrm{o}$ refers to the angular frequency components in the spatial dimension. Figure 3.2 gives a graphical intuition of these frequency components. The advantage of the convention $m_\mathrm{s} - m_\mathrm{o}$ for spatial-angular frequency components, instead of m_s, is that rotation of $\mathcal{S}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}$ is achieved by multiplication with $e^{-im_\mathrm{o}\theta}$, i.e. only dependent on m_o. This convention is also more convenient when considering orientationally bandlimited orientation scores (see Subsection 3.5.3) and it is more consistent with the Fourier transform on $SE(2)$ (see Section 3.6).

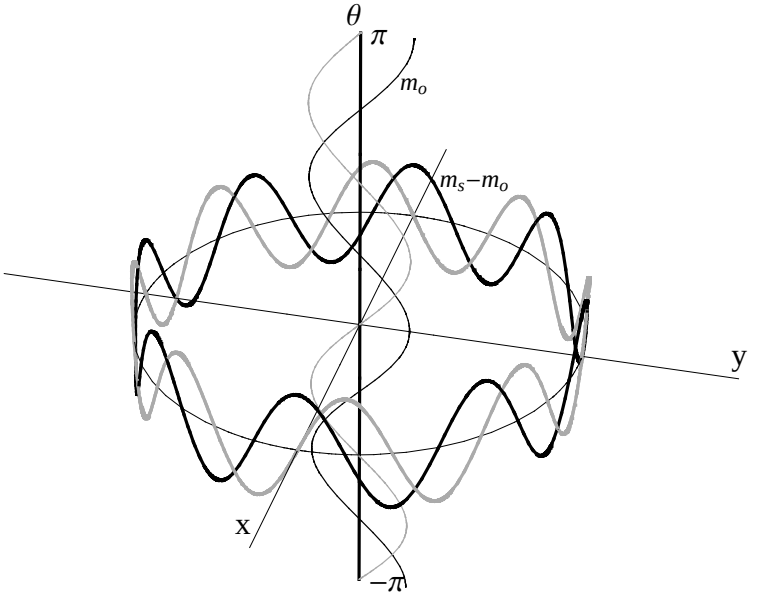


Figure 3.2: Schematic view of the orientation-angular and spatial-angular frequency components of an $SE(2)$-kernel. The orientation-angular frequency components represent the frequencies in the θ-dimension and are indexed by m_o. The spatial-angular frequency components represent the frequency components over a circle in the x, y plane. Note that in our convention, the latter frequency components are indexed by $m_\mathrm{s} - m_\mathrm{o}$ rather than m_s, i.e. the indexation is different for each fixed value of orientation-angular frequency m_o. In the picture we just display 2 arbitrary frequency components (namely $m_\mathrm{o} = 2$ and $m_\mathrm{s} - m_\mathrm{o} = 8$), where the black and gray curves show the cosine and sine parts respectively.

Next, we also define the *orientation-angular Fourier decomposition* $\mathcal{F}_\theta : (SE(2) \to \mathbb{C}) \to (\mathbb{Z} \times \mathbb{R}^+ \times \mathbb{T} \to \mathbb{C})$:

$$\mathcal{F}_\theta[\Psi]_{m_\mathrm{o}}(r,\phi) = \mathcal{F}_\mathbb{T}[\Psi(r,\phi,\cdot)]_{m_\mathrm{o}} = \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} \Psi(r,\phi,\theta) e^{-i m_\mathrm{o}\theta} \mathrm{d}\theta, \tag{3.42}$$

$$\mathcal{F}_\theta^{-1}[\hat{\Psi}](r,\phi,\theta) = \frac{1}{\sqrt{2\pi}} \sum_{m_\mathrm{o}\in\mathbb{Z}} \hat{\Psi}_{m_\mathrm{o}}(r,\phi) e^{i m_\mathrm{o}\theta}. \tag{3.43}$$

The reason to define $\mathcal{F}_\theta$ is to avoid the notational clutter that is needed to specify on which argument of a function on $SE(2)$ the angular Fourier decomposition needs to be performed.

The basis functions that we will use to steer the $SE(2)$-convolution are defined by $\mathcal{C}[\Psi] : \mathbb{Z} \times \mathbb{Z} \times \mathbb{R}^2 \to \mathbb{C}$

$$\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}} \mathcal{S}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}(r) e^{i(m_\mathrm{s}-m_\mathrm{o})\phi} \quad \text{with } \boldsymbol{x} = (r\cos\phi, r\sin\phi). \tag{3.44}$$

These basis functions will further on be called *steerable components*. The relation with $\mathcal{F}_\theta[\Psi]$ cf. (3.42) is given by

$$\sum_{m_\mathrm{s}\in\mathbb{Z}} \mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}(\boldsymbol{x}) = \mathcal{F}_\theta[\Psi]_{m_\mathrm{o}}(\boldsymbol{x}), \tag{3.45}$$

and the kernel Ψ is obtained by

$$\Psi(\boldsymbol{x},\theta) = \frac{1}{\sqrt{2\pi}} \sum_{m_\mathrm{s}\in\mathbb{Z}} \sum_{m_\mathrm{o}\in\mathbb{Z}} \mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}(\boldsymbol{x}) e^{i m_\mathrm{o}\theta}. \tag{3.46}$$

A rotated kernel $\mathcal{R}_\alpha[\Psi]$ is constructed by

$$\mathcal{R}_\alpha[\Psi](\boldsymbol{x},\theta) = \frac{1}{\sqrt{2\pi}} \sum_{m_\mathrm{s}\in\mathbb{Z}} e^{-i m_\mathrm{s}\alpha} \sum_{m_\mathrm{o}\in\mathbb{Z}} \mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}(\boldsymbol{x}) e^{i m_\mathrm{o}\theta}. \tag{3.47}$$

Kernel Ψ is steerable if the number of components in both sums is finite. For notational simplicity, we will write $\sum_{m\in\mathbb{Z}}$, leaving out the explicit specification of the angular frequency bounds.

To make the $SE(2)$-convolution steerable we insert (3.47) into (3.17), yielding

$$\begin{aligned}
&(\Psi *_{SE(2)} U)(\boldsymbol{x}, \theta) \\
&= \int_{\mathbb{R}^2} \int_0^{2\pi} \left(\frac{1}{\sqrt{2\pi}} \sum_{m_\mathrm{s} \in \mathbb{Z}} e^{-i m_\mathrm{s} \theta'} \sum_{m_\mathrm{o} \in \mathbb{Z}} \mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}(\boldsymbol{x} - \boldsymbol{x}') e^{i m_\mathrm{o} \theta} \right) U(\boldsymbol{x}', \theta') \mathrm{d}\theta' \mathrm{d}\boldsymbol{x}' \\
&= \sum_{m_\mathrm{o} \in \mathbb{Z}} e^{i m_\mathrm{o} \theta} \sum_{m_\mathrm{s} \in \mathbb{Z}} \int_{\mathbb{R}^2} \mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}(\boldsymbol{x} - \boldsymbol{x}') \underbrace{\left(\int_0^{2\pi} \frac{1}{\sqrt{2\pi}} e^{-i m_\mathrm{s} \theta'} U(\boldsymbol{x}', \theta') \mathrm{d}\theta' \right)}_{\mathcal{F}_\theta[U]_{m_\mathrm{s}}(\boldsymbol{x}')} \mathrm{d}\boldsymbol{x}' \\
&= \sum_{m_\mathrm{o} \in \mathbb{Z}} e^{i m_\mathrm{o} \theta} \sum_{m_\mathrm{s} \in \mathbb{Z}} \left(\mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}} \right)(\boldsymbol{x}) \\
&= \sqrt{2\pi}\, \mathcal{F}_\theta^{-1} \left[\sum_{m_\mathrm{s} \in \mathbb{Z}} \left(\mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}} \right)(\boldsymbol{x}) \right] (\theta).
\end{aligned} \tag{3.48}$$

This derivation shows that we can convert an $SE(2)$-convolution into a sum of ordinary 2D convolutions. This is beneficial for implementations, since it is easier to implement and no interpolations need to be performed in contrast to the method of Subsection 3.3.3. Furthermore it is computationally more efficient, depending on the number of nonzero components in $\sum_{m_\mathrm{o} \in \mathbb{Z}}$ and $\sum_{m_\mathrm{s} \in \mathbb{Z}}$, as discussed in the next subsection. By applying $\mathcal{F}_\theta$ on both sides, we can rewrite the latter equation as

$$\mathcal{F}_\theta[\Psi *_{SE(2)} U]_{m_\mathrm{o}}(\boldsymbol{x}) = \sqrt{2\pi} \sum_{m_\mathrm{s} \in \mathbb{Z}} \left(\mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}} \right)(\boldsymbol{x}), \tag{3.49}$$

We can also use the standard convolution theorem on $\mathbb{R}^2$ to rewrite equation (3.48) using 2D Fourier transforms

$$\begin{aligned}
&(\Psi *_{SE(2)} U)(\boldsymbol{x}, \theta) \\
&\quad = (2\pi)^{\frac{3}{2}}\, \mathcal{F}_\theta^{-1} \left[\mathcal{F}_{\mathbb{R}^2}^{-1} \left[\sum_{m_\mathrm{s} \in \mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}]\, \mathcal{F}_{\mathbb{R}^2}[\mathcal{F}_\theta[U]_{m_\mathrm{s}}] \right] (\boldsymbol{x}) \right] (\theta),
\end{aligned} \tag{3.50}$$

where we note that using equation (3.15) and (3.44) we can write

$$\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}](\rho \cos\varphi, \rho \sin\varphi) = \sqrt{2\pi}\, \mathcal{H}_{m_\mathrm{s} - m_\mathrm{o}}[\mathcal{S}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}](\rho) e^{i(m_\mathrm{s} - m_\mathrm{o})\varphi}. \tag{3.51}$$

The latter equation is useful if we have an analytic expressions for $\mathcal{S}[\Psi]_{m_\mathrm{o}, m_\mathrm{s}}$ for which an analytical Hankel transform exists. In that case we can directly sample the steerable components in the Fourier domain.

3.5.2 Implementation of Steerable $SE(2)$-Convolution

In this subsection we will discuss possible implementations of the steerable $SE(2)$-convolution and their complexities. In the estimates for algorithmic complexity, we will not take into account the time it takes to numerically calculate the steerable components $\mathcal{C}[\Psi]_{m_o,m_s}$ from Ψ, because for many kernels it is possible to find these expressions analytically. Furthermore, in many practical cases multiple $SE(2)$-convolutions with the same kernel have to be applied, so it only needs to be calculated once.

From (3.48) it is straightforward to implement the steerable $SE(2)$convolution. Let C_s denote the number of values of m_s for which $\mathcal{C}[\Psi]_{m_o,m_s}$ is nonzero and let C_o denote the number of values of m_o for which $\mathcal{C}[\Psi]_{m_o,m_s}$ is nonzero. Given that we have all steerable components $\mathcal{C}[\Psi]_{m_o,m_s}$ at hand, the algorithm consists of the following steps

1. FFT in orientation dimension on U, i.e. $U \mapsto \mathcal{F}_\theta[U]_{m_s}$ cf. (3.42): complexity $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$;
2. For each nonzero steerable component $\mathcal{C}[\Psi]_{m_o,m_s}$ (i.e. $C_o C_s$ terms) perform a spatial $\mathbb{R}^2$-convolution and sum over m_s, cf. (3.49): $\mathcal{O}\left(C_o C_s N_s^2 K_s^2\right)$;
3. Inverse FFT in orientation dimension on the result, cf. (3.43): $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$.

The $C_o C_s$ convolutions on $\mathbb{R}^2$ have the highest complexity, so the total complexity is given by $\mathcal{O}\left(C_o C_s N_s^2 K_s^2\right)$.

For larger spatial kernels it is more efficient to perform the $\mathbb{R}^2$-convolutions via the Fourier domain, cf. (3.50). In that case the algorithm consists of the following steps

1. 3D FFT of U, i.e. $U \mapsto \mathcal{F}_{\mathbb{R}^2}[\mathcal{F}_\theta[U]_{m_s}])$: complexity $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$, assuming $N_s \geq N_o$;
2. For each fourier-transformed nonzero steerable component $\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_o,m_s}]$ (i.e. $C_o C_s$ terms) perform a multiplication for each spatial frequency component (N_s^2 terms) and sum over m_s: $\mathcal{O}\left(C_o C_s N_s^2\right)$;
3. 3D inverse FFT on the result, i.e. $\mathcal{F}_\theta^{-1} \circ \mathcal{F}_{\mathbb{R}^2}^{-1}$: $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$.

The total complexity in this case is $\mathcal{O}\left(N_s^2(N_o \log N_s + C_o C_s)\right)$, where the dominant term depends on the values of $N_o \log N_s$ and $C_o C_s$.

An overview of complexities for the different implementations is given in Table 3.2. Clearly, the non-steerable (described in Subsection 3.3.3) and steerable

FFT implementations described in this subsection have the lowest computational complexity. Besides the standard reduction achieved by FFT, an additional reduction in complexity is achieved since the FFTs on U can be applied once instead of for each summand in (3.32) resp. (3.50). Which of the two FFT implementations is most efficient depends on the terms $N_o K_o$ resp $C_o C_s$. If $C_o = N_o$ and $C_s = N_o$ and the number of samples in orientation dimension $K_o < N_o$, the non-steerable implementation will be faster, while if the number of steerable components C_o are C_s are limited, the steerable implementation will be faster.

If one does not have an analytical expression for the steerable components $\mathcal{C}[\Psi]_{m_o,m_s}$, once can use the following numerical recipe to obtain these components from a Cartesian sampled kernel $\Psi(x, y, l)$

1. Polar resampling using some interpolation scheme, leading to a sampled $\Psi(r, \phi, l)$ where r and ϕ are equidistantly sampled an $\boldsymbol{x} = (r\cos\phi, r\sin\phi)$;
2. Apply 1D FFTs in the orientation dimension, padded with zeroes to have N_o samples, to obtain $\mathcal{F}_\theta[\Psi]_{m_o}$ cf. (3.42);
3. Apply 1D FFTs in the ϕ dimension to obtain $\mathcal{S}[\Psi]_{m_o,m_s}$ cf. (3.40);
4. Sample each nonzero $\mathcal{S}[\Psi]_{m_o,m_s}(r)e^{i(m_s-m_o)\phi}$ on the Cartesian grid to obtain $\mathcal{C}[\Psi]_{m_o,m_s}$ cf. (3.44).

The complexity of these numerical calculations is $\mathcal{O}\left(K_s^2 N_o I\right) + \mathcal{O}\left(K_s^2 N_o \log N_o\right) + \mathcal{O}\left(C_o C_s K_s^2\right)$ where I denotes the complexity of a single interpolation. In case one needs $\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_o,m_s}]$ the recipe becomes

1. Perform step 1, 2, and 3 of the previous recipe;
2. Apply the Hankel transform in r direction (padded with zeroes to have N_s samples) to obtain $\mathcal{H}_{m_s-m_o}[\mathcal{S}[\Psi]_{m_o,m_s}]$ cf. (3.13);
3. Sample each nonzero $\mathcal{H}_{m_s-m_o}[\mathcal{S}[\Psi]_{m_o,m_s}](\rho)e^{i(m_s-m_o)\varphi}$ on the Cartesian frequency grid $\boldsymbol{\omega} = \rho(\cos\varphi, \sin\varphi)$ to obtain $\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_o,m_s}]$.

The complexity of these numerical calculations is $\mathcal{O}\left(K_s^2 N_o I\right) + \mathcal{O}\left(K_s^2 N_o \log N_o\right) + \mathcal{O}\left(C_o C_s N_s^2\right)$ where I denotes the complexity of a single interpolation.

3.5.3 Considerations on Sampling the Orientation Dimension

Since we sample the orientation dimension using N_o equidistant samples, the sampling theorem (or Nyquist theorem) states that the orientation-angular frequency components $\mathcal{F}_\theta[U]_{m_o}$ are bandlimited by $M_o \leq N_o/2$. This naturally

poses requirements on both the orientation score construction kernel ψ and the $SE(2)$-kernel Ψ, which we will explain in the following subsections.

3.5.3.1 Sampling Theorem on Orientation Score Construction Kernels

Remember from Subsection 3.4.1 how we can create steerable kernels $\mathbb{R}^2 \to \mathbb{C}$. If we decompose the wavelet construction kernel ψ in $2M+1$ steerable components ψ_m, the orientation score construction formula (2.6) becomes

$$U_f(\boldsymbol{x}, \theta) = \sum_{m=-M}^{M} e^{-im\theta}\big((\psi_m * f)(\boldsymbol{x})\big). \tag{3.52}$$

From the latter equation we observe that M determines the orientation-angular bandwidth (due to the term $e^{-im\theta}$). Therefore, ψ should be a steerable filter in "the classical sense" (Subsection 3.4.1) with spatial-angular bandwidth $M \leq N_o/2$. If we ignore this requirement one can still construct (invertible) orientation scores, but one might observe aliasing in orientation columns because $e^{-im\theta}$ for $m > N_o/2$ cannot be appropriately sampled[2].

Note that a steerable kernel ψ with spatial-angular bandwidth M ensures that $\mathcal{F}_\theta[U_f]_{m_\mathrm{o}} = \mathcal{F}_\mathbb{T}[U_f(r,\phi,\cdot)]_{m_\mathrm{o}}$ is nonzero only for $|m_\mathrm{o}| \leq M$, i.e. orientation-angularly bandlimited. However, it does *not* ensure that $\mathcal{F}_\mathbb{T}[U_f(r,\cdot,\theta)]_{m_\mathrm{s}-m_\mathrm{o}}$ is band-limited, i.e. spatial-angular bandlimited. If the spatial-angular frequency components of f, $\mathcal{F}_\mathbb{T}[f(r,\cdot)]_k$, are band-limited by $|k| \leq K$, then using (3.16) we find that $|m_\mathrm{s} - m_\mathrm{o}| \leq K + M$.

3.5.3.2 Sampling Theorem on $SE(2)$-Kernels

We can also find bounds for the angular bandwidth of an $SE(2)$-kernel. The $SE(2)$-kernel should not contain frequencies above these bounds, because the truncation of frequencies due to the sampling will cause aliasing. Suppose that the input orientation score U is orientation-angularly bandlimited by $|m_\mathrm{o}| \leq M_\mathrm{o} \leq N_o/2$ and that the resulting orientation score should have the same bandwidth, which makes sense if we take the same number of orientations for the input and the output. Then, from (3.50) we get two bounds for the bandwidth of Ψ:

$$-M_\mathrm{o} \leq m_\mathrm{o} \leq M_\mathrm{o}, \quad \text{and} \quad -M_\mathrm{o} \leq m_\mathrm{s} \leq M_\mathrm{o}, \tag{3.53}$$

where the left inequality follows from the bandwidth of the result $\Psi *_{SE(2)} U$ and the right inequality follows from the bandwidth of the input U. This important result means that, according to the sampling theorem in the orientation dimension,

[2]It should be noted that the kernels ψ defined in Subsection 2.4.1 are not exactly steerable, however by choosing the right parameters the responses for $m > N_o/2$ are ensured to be relatively small so this does not lead to significant aliasing problems.

all $SE(2)$-kernels *should* be steerable kernels with $|m_{\mathrm{o}}| \leq N_o/2$ and $|m_{\mathrm{s}}| \leq N_o/2$.

In case of an analytic kernel Ψ which is not steerable by itself, i.e. not angularly bandlimited, the question is whether it is more reliable to use a steerable or a non-steerable $SE(2)$-convolution. The complexities of both approaches are the same if we take a steerable $SE(2)$-convolution with $M_{\mathrm{s}} = M_{\mathrm{o}} = N_o/2$. However, given the fact that we have sampled the orientation dimension and given the fact that the kernel (for example Figure 3.3(a)) is not angularly bandlimited, the steerable implementation with truncated steerable components will result in Gibbs oscillations on anisotropic signals of the form $U(x, y, \theta) = \delta(x)\delta(y)\delta(\theta)$ (see Figure 3.3(b)), while the response on isotropic signals of the form $U(x, y, \theta) = \delta(x)\delta(y)$ will be perfectly isotropic (see Figure 3.3(d)).

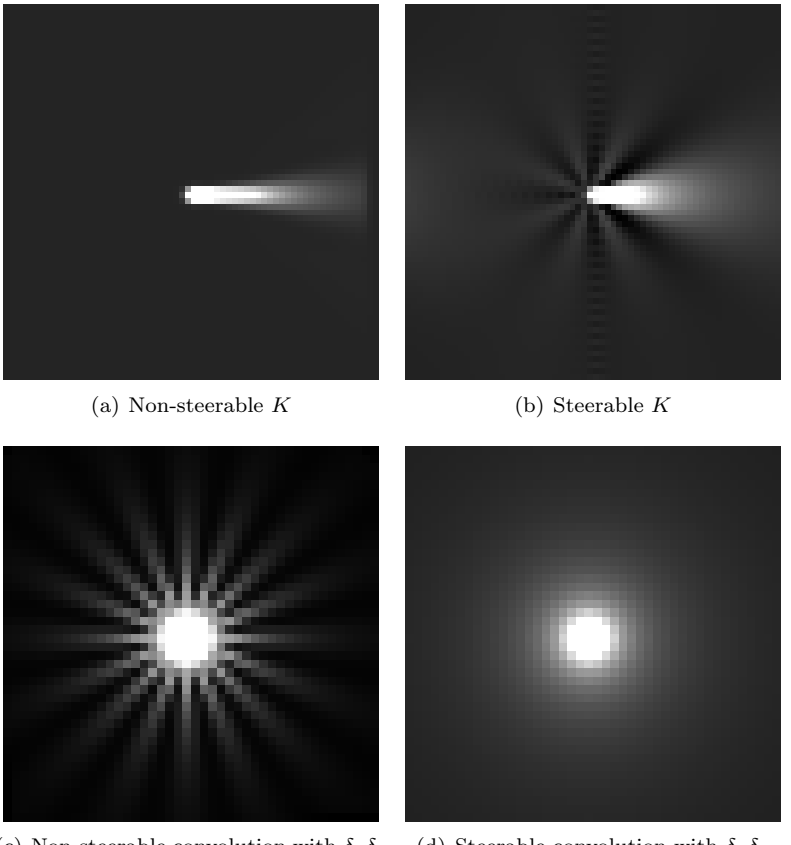
On the other hand, when "naively" sampling the $SE(2)$-kernel and using a non-steerable implementation, for signals $U(x, y, \theta) = \delta(x)\delta(y)\delta(\theta)$ no Gibbs artefacts will occur (see Figure 3.3(a)). However, the response on a signal $U(x, y, \theta) = \delta(x)\delta(y)$ will not be isotropic at all: we observe aliasing artefacts since we sample angular frequencies above the Nyquist bound (see Figure 3.3(c)). In conclusion, it is in principle preferable to use the steerable implementation, in order to have full "control" over the occurrence of Gibbs oscillations and aliasing artefacts.

3.6 Steerable $SE(2)$-Convolution and the Fourier Transform on $SE(2)$

In this section we will introduce the Fourier transform on $SE(2)$ and describe how it is implemented in [22]. Similar to convolutions on $\mathbb{R}^n$, $SE(2)$-convolutions can be computed using the $SE(2)$-Fourier transform and the convolution theorem on $SE(2)$ that will be provided in this subsection. We will study how this approach relates to steerable $SE(2)$-convolution. By explicitly rewriting the convolution theorem on $SE(2)$ to the expression for steerable $SE(2)$-convolution cf. (3.50), we make apparent what the difference between the two approaches is, showing that the steerable $SE(2)$-convolution is more efficient since unnecessary calculation steps are omitted.

3.6.1 The Fourier Transform on $SE(2)$

As mentioned in Subsection 3.4.1, the irreducible representations of $SE(2)$ are multi-dimensional. In principle they are even infinite-dimensional, but if one assumes a certain orientation-angular and spatial-angular bandlimit they become finite-dimensional and can be represented as matrices. Expressing group elements



(a) Non-steerable K (b) Steerable K

(c) Non-steerable convolution with $\delta_x\delta_y$ (d) Steerable convolution with $\delta_x\delta_y$

Figure 3.3: Illustration of Gibbs oscillations and aliasing artefacts in orientation scores. (a) Shows the θ-marginal of the stochastic completion kernel cf. (3.78), i.e. $\int_0^{2\pi} K_{\sigma,\lambda}(\boldsymbol{x},\theta)\mathrm{d}\theta$, with $\sigma = 0.03$ and $\lambda = 0.02$, sampled on a 64×64 grid. To make this kernel steerable one needs to truncate the steerable components, which is shown in (b) for the case $M_{\mathrm{o}} \leq 7$. Clearly, *Gibbs oscillations* occur due to the truncation of the angular frequencies. Notice that a non-steerable $SE(2)$ convolution of (a) with $\delta_x\delta_y\delta_\theta$ renders (a) again, while a steerable $SE(2)$ convolution of (b) with $\delta_x\delta_y\delta_\theta$ renders (b) again. (c) Shows the θ-marginal of the non-steerable convolution of (a) with an "isotropic" orientation score $\delta_x\delta_y$ sampled with 16 orientations and spatial size 44×44. In theory, this should render a perfectly isotropic result. However, since kernel (a) is not angularly bandlimited, Equation (3.53) is not satisfied, giving rise to *aliasing artefacts*. For the steerable convolution with $M_{\mathrm{o}} \leq 7$ equation (3.53) *is* satisfied, rendering the isotropic result of (d).

in polar coordinates, i.e. $g = (r, \phi, \theta)$, the unitary irreducible matrix representations of $SE(2)$ are given by (following [22, Section 10.2][3])

$$\Theta^{\rho}_{m_{\mathrm{s}} m_{\mathrm{o}}}(r, \phi, \theta) = \frac{(-i)^{m_{\mathrm{o}} - m_{\mathrm{s}}}}{2\pi} e^{i(m_{\mathrm{s}} - m_{\mathrm{o}})\phi + i m_{\mathrm{o}} \theta} J_{m_{\mathrm{o}} - m_{\mathrm{s}}}(\rho r), \tag{3.54}$$

where $\rho \in \mathbb{R}^+$ is identified with the radius in the Fourier domain ($\rho = \|\boldsymbol{\omega}\|$) and "enumerates" the matrices of irreducible representations. The indices m_{s} and m_{o} enumerate the columns resp. rows of these matrices. In the notation $\Theta^{\rho}_{m_{\mathrm{s}} m_{\mathrm{o}}}$, ρ is just an upper index and should not be confused with taking a power ρ.

Following [22], the Fourier transform on $SE(2)$ is given by

$$\mathcal{F}_{SE(2)}[\Psi]_{m_{\mathrm{o}}, m_{\mathrm{s}}}(\rho) = \int_{SE(2)} \Psi(g) \Theta^{\rho}_{m_{\mathrm{o}}, m_{\mathrm{s}}}(g^{-1}) \mathrm{d}g. \tag{3.55}$$

The inverse Fourier transform on $SE(2)$ of $\hat{\Psi} = \mathcal{F}_{SE(2)}[\Psi]$ is given by

$$\mathcal{F}^{-1}_{SE(2)}(\hat{\Psi}) = \int_0^{\infty} \sum_{m_{\mathrm{o}} \in \mathbb{Z}} \sum_{m_{\mathrm{s}} \in \mathbb{Z}} (\hat{\Psi}(\rho))_{m_{\mathrm{o}}, m_{\mathrm{s}}} (\Theta^{\rho}_{m_{\mathrm{o}}, m_{\mathrm{s}}}(g)) \rho \, \mathrm{d}\rho. \tag{3.56}$$

By inserting the expression for the irreducible representations in (3.55) and (3.56) and using the relations $g^{-1} = (r, \phi, \theta)^{-1} = (r, \phi + \pi - \theta, \theta)$ and $J_{-m}(\rho r) = (-1)^m J_m(\rho r)$, we find as explicit expression of the Fourier transform on $SE(2)$

$$\begin{aligned} &\mathcal{F}_{SE(2)}[\Psi]_{m_{\mathrm{o}}, m_{\mathrm{s}}}(\rho) \\ &= \frac{i^{m_{\mathrm{s}} - m_{\mathrm{o}}}}{2\pi} \int_0^{\infty} \int_0^{2\pi} \int_0^{2\pi} \Psi(r, \phi, \theta) e^{-i(m_{\mathrm{s}} - m_{\mathrm{o}})\phi - i m_{\mathrm{o}} \theta} J_{m_{\mathrm{s}} - m_{\mathrm{o}}}(\rho r) r \, \mathrm{d}r, \end{aligned} \tag{3.57}$$

and the inverse is given by

$$\begin{aligned} &\mathcal{F}^{-1}_{SE(2)}[\hat{\Psi}](r, \phi, \theta) \\ &= \sum_{m_{\mathrm{o}} \in \mathbb{Z}} \sum_{m_{\mathrm{s}} \in \mathbb{Z}} \frac{(-i)^{m_{\mathrm{s}} - m_{\mathrm{o}}}}{2\pi} \int_0^{\infty} \hat{\Psi}_{m_{\mathrm{o}}, m_{\mathrm{s}}}(\rho) e^{i(m_{\mathrm{s}} - m_{\mathrm{o}})\phi + i m_{\mathrm{o}} \theta} J_{m_{\mathrm{s}} - m_{\mathrm{o}}}(\rho r) \rho \, \mathrm{d}\rho. \end{aligned} \tag{3.58}$$

3.6.2 Implementation of a Fast $SE(2)$-Fourier Transform

Chirikjian [22, Section 11.2] proposes to implement the Fourier transform on $SE(2)$ of an orientation score $U(\boldsymbol{x}, \theta)$ using the following steps

[3] where we use a slightly different convention concerning factors and minus signs to make it consistent with the rest of this chapter.

1. Fourier transform (FFT) in the spatial plane followed by resampling (by interpolation) the Fourier domain in polar coordinates, i.e. numeric calculation of

$$\hat{U}(\rho,\varphi,\theta) = \frac{1}{2\pi}\int_{\mathbb{R}^2} U(x,y,\theta)e^{i(x\rho\cos\varphi+y\rho\sin\varphi)}dx\,dy, \tag{3.59}$$

   where ρ is the irreducible representation, which coincides with a circle with radius ρ in the spatial Fourier domain, and φ is the angle in the spatial Fourier domain.

2. Apply orientation-angular Fourier decomposition using FFT

$$\hat{U}_{m_o}(\rho,\varphi) = \frac{1}{\sqrt{2\pi}}\int_0^{2\pi} \hat{U}(\rho,\varphi,\theta)e^{im_o\theta}\mathrm{d}\theta. \tag{3.60}$$

3. Apply spatial-angular Fourier decomposition using FFT

$$\mathcal{F}_{SE(2)}[U]_{m_o,m_s}(\rho) = \frac{1}{\sqrt{2\pi}}\int_0^{2\pi}\left[\hat{U}_{m_o}(\rho,\varphi)e^{-im_s\varphi}\right]e^{im_s\varphi}\mathrm{d}\varphi. \tag{3.61}$$

The inverse transformation is simply achieved by reversing all steps. The complexity is $\mathcal{O}\left(N_o N_s^2 \log N_s\right) + \mathcal{O}\left(N_o N_s^2 I\right)$ where I denotes the complexity of a single interpolation. If B-spline interpolation is used $I = k^2$ where k is the spline order, while if the more accurate Fourier interpolation is used then $I = (\log N_s)^2$, see [22, Section 11.1].

3.6.3 $SE(2)$-Convolution Implementation using Fast $SE(2)$-Fourier Transforms

Following [22, Section 10.4] the *convolution theorem* on $SE(2)$ is given by

$$\mathcal{F}_{SE(2)}[\Psi *_{SE(2)} U]_{m_o,m_s}(\rho) = (2\pi)^2 \sum_{n\in\mathbb{Z}} \mathcal{F}_{SE(2)}[\Psi]_{m_o,n}(\rho)\mathcal{F}_{SE(2)}[U]_{n,m_s}(\rho). \tag{3.62}$$

The $SE(2)$-convolution can be computed using this equation. The implementation involves $SE(2)$-Fourier transforms of Ψ and U using the algorithm described in the previous subsection, followed by multiplication in the $SE(2)$-Fourier domain, which involves N_s^2 *matrix multiplications* of matrices with size ${N_o}^2$, rather than scale multiplications. Finally, the inverse $SE(2)$-Fourier transform is applied to get the result of the $SE(2)$-convolution.

Matrix multiplication has a complexity of $\mathcal{O}\left({N_o}^{\zeta}\right)$ with $2 \leq \zeta \leq 3$, that is, a naive implementation yields $\zeta = 3$ but algorithms exist that are asymptotically faster such as the Strassen algorithm [128] with $\zeta = \log_2 7$, and the Coppersmith–Winograd algorithm [25] with $\zeta = 2.376$. These algorithms are only faster on larger matrices, depending on the implementation and hardware architecture. The lower bound $2 \leq \zeta$ is only a theoretical lower bound since this is the least complexity needed to iterate through all matrix elements.

Therefore, the multiplication step in (3.62) has a complexity of $\mathcal{O}\left(N_s^2 {N_o}^{\zeta}\right)$ with $2 \leq \zeta \leq 3$. In case of spline interpolation the total complexity becomes $\mathcal{O}\left(N_s^2(N_o \log N_s + {N_o}^{\zeta})\right)$, where the spline order k is not taken into account as it is not relevant for this complexity estimate. Given that $C_s = C_o = N_o$, or assuming that computations are not performed for coefficients of $\mathcal{F}_{SE(2)}[U]_{m_o,m_s}$ that are known to be zero, this is the same complexity as for the steerable $SE(2)$-convolution approach cf. Subsection 3.5.2. The total complexity with Fourier interpolation becomes $\mathcal{O}\left(N_s^2\left(N_o(\log N_s)^2 + {N_o}^{\zeta}\right)\right)$, which is higher than the complexity for steerable $SE(2)$-convolution. All complexity estimates are tabelized in Table 3.2.

3.6.4 From $SE(2)$-Fourier to Steerable $SE(2)$-Convolution

In this subsection we will rewrite the latter equation (3.62) to a steerable $SE(2)$-convolution cf. (3.50), which will make the relation between the two apparent.

First, we observe that the $SE(2)$-Fourier transform cf. (3.57) can be rewritten as a concatenation of the $\mathcal{S}[\Psi]$ operation cf. (3.40) and the Hankel transform cf. (3.13), i.e.

$$\mathcal{F}_{SE(2)}[\Psi]_{m_o,m_s}(\rho) = \mathcal{H}_{m_s-m_o}\left[\mathcal{S}[\Psi]_{m_o,m_s}\right](\rho), \tag{3.63}$$

which, by multiplying with $e^{i(m_s-m_o)\varphi}$ and using (3.51), can be rewritten as

$$\mathcal{F}_{SE(2)}[\Psi]_{m_o,m_s}(\rho)e^{i(m_s-m_o)\varphi} = \frac{1}{\sqrt{2\pi}}\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_o,m_s}](\rho\cos\varphi, \rho\sin\varphi). \tag{3.64}$$

If we multiply both sides of (3.62) with $e^{i(m_s-m_o)\varphi}$ and use the latter equation (3.64), we obtain

$$\begin{aligned}&\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi *_{SE(2)} U]_{m_o,m_s}](\boldsymbol{\omega}) \\ &\qquad = (2\pi)^{\frac{3}{2}} \sum_{n\in\mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_o,n}](\boldsymbol{\omega})\, \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[U]_{n,m_s}](\boldsymbol{\omega}),\end{aligned} \tag{3.65}$$

with $\boldsymbol{\omega} = (\rho\cos\varphi, \rho\sin\varphi)$.

Next, we sum both sides of (3.65) over index variable m_{s},

$$\sum_{m_{\mathrm{s}}\in\mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi *_{SE(2)} U]_{m_{\mathrm{o}},m_{\mathrm{s}}}](\boldsymbol{\omega}) = (2\pi)^{\frac{3}{2}} \sum_{n\in\mathbb{Z}} \left(\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_{\mathrm{o}},n}](\boldsymbol{\omega}) \sum_{m_{\mathrm{s}}\in\mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[U]_{n,m_{\mathrm{s}}}](\boldsymbol{\omega}) \right), \quad (3.66)$$

and we apply equation (3.45) on both the left-hand side of the latter equation as well as the right part of the the right-hand side, yielding

$$\mathcal{F}_{\mathbb{R}^2}[\mathcal{F}_\theta[\Psi *_{SE(2)} U]_{m_{\mathrm{o}}}](\boldsymbol{\omega}) = (2\pi)^{\frac{3}{2}} \sum_{n\in\mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_{\mathrm{o}},n}](\boldsymbol{\omega})\, \mathcal{F}_{\mathbb{R}^2}[\mathcal{F}_\theta[U]_n](\boldsymbol{\omega}), \quad (3.67)$$

Applying $\mathcal{F}^{-1}_{\mathbb{R}^2}$ and $\mathcal{F}^{-1}_{\theta}$ on both sides yields

$$(\Psi *_{SE(2)} U)(\boldsymbol{x},\theta) = (2\pi)^{\frac{3}{2}}\, \mathcal{F}^{-1}_{\theta} \left[\mathcal{F}^{-1}_{\mathbb{R}^2} \left[\sum_{n\in\mathbb{Z}} \mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[\Psi]_{m_{\mathrm{o}},n}]\, \mathcal{F}_{\mathbb{R}^2}[\mathcal{F}_\theta[U]_n] \right](\boldsymbol{x}) \right](\theta), \quad (3.68)$$

which is exactly the same equation as the steerable $SE(2)$-convolution cf. (3.50).

Comparing an $SE(2)$-convolution implementation that uses (a) the convolution theorem approach cf. Subsection 3.6.3 and (b) the steerable $SE(2)$-convolution cf. (3.50) implementation, we observe that in approach (a) one applies full-angular Fourier decomposition on orientation score U, which amounts to step 1 and step 3 of the $SE(2)$-Fourier algorithm in Subsection 3.6.2, and to obtain the final result this operation has to be inverted. These steps are omitted in (b), leading to a reduction in the number of computations and a reduction in complexity, see Table 3.2. Furthermore, the fact that the Cartesian to polar grid interpolation and vice versa is not needed in case of steerable $SE(2)$-convolution is an advantage.

3.7 Special Cases of $SE(2)$-convolution

In this section we will consider useful special cases of $SE(2)$-convolutions. Each of these special cases leads to a simplification of the (steerable) $SE(2)$-convolution that can be exploited to gain efficiency.

3.7.1 π-Periodic Orientation Scores

If one is only interested in segmentation or enhancement of 180°-symmetric structures, such as lines and / or ridges, it is sufficient to consider orientation scores

with a period π rather than 2π. In this case, the orientation-angular Fourier decomposition $\mathcal{F}_\theta[U]_{m_\mathrm{o}}$ is zero for all odd m_o. Consequently, by considering equation (3.49), we see that the steerable components $\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}$ are irrelevant for all odd values of m_s, so one should restrict to kernels with $\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} = 0$ for all odd values of m_s. If the result is also required to be π-periodic then the steerable components must also be zero for all odd values of m_o.

In conclusion, for π-periodic steerable $SE(2)$-convolutions, we only have nonzero *even* orientation-angular and spatial-angular frequency components, which leads to a gain in speed between a factor 2 and 4, since N_o, C_o, and C_s are halved.

3.7.2 Real-valued Orientation Scores and $SE(2)$-Kernels

In case of real-valued orientation orientation scores U we have the relation

$$U = \overline{U} \;\rightarrow\; \mathcal{F}_\theta[U]_{m_\mathrm{o}} = \overline{\mathcal{F}_\theta[U]_{-m_\mathrm{o}}}. \tag{3.69}$$

For real-valued steerable components $\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}}$ we have the relations (cf. (3.44))

$$\mathcal{S}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} = \overline{\mathcal{S}[\Psi]_{-m_\mathrm{o},-m_\mathrm{s}}}, \quad \text{so} \quad \mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} = \overline{\mathcal{C}[\Psi]_{-m_\mathrm{o},-m_\mathrm{s}}}. \tag{3.70}$$

Using these relations we can rewrite the steerable $SE(2)$-convolution of (3.48) as

$$\begin{aligned}(\Psi *_{SE(2)} U)(\boldsymbol{x},\theta) &= \sum_{m_\mathrm{o}=-M_\mathrm{o}}^{M_\mathrm{o}} e^{im_\mathrm{o}\theta} \sum_{m_\mathrm{s}=-M_\mathrm{s}}^{M_\mathrm{s}} \big(\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}}\big)(\boldsymbol{x}) \\ &= \sum_{m_\mathrm{o}=0}^{M_\mathrm{o}} e^{im_\mathrm{o}\theta} \sum_{m_\mathrm{s}=-M_\mathrm{s}}^{M_\mathrm{s}} \Big(\big(\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}}\big)(\boldsymbol{x}) \\ &\qquad + (1-\delta_{m_\mathrm{o}})\big(\mathcal{C}[\Psi]_{-m_\mathrm{o},-m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{-m_\mathrm{s}}\big)(\boldsymbol{x})\Big).\end{aligned} \tag{3.71}$$

Using (3.69) and (3.70) and the property $A{*}B + \overline{A}{*}\overline{B} = 2\,\mathrm{Re}(A{*}B)$ we obtain

$$\begin{aligned}&(\Psi *_{SE(2)} U)(\boldsymbol{x},\theta) \\ &= \mathrm{Re}\left(\sum_{m_\mathrm{o}=0}^{M_\mathrm{o}} (2-\delta_{m_\mathrm{o}}) e^{im_\mathrm{o}\theta} \sum_{m_\mathrm{s}=-M_\mathrm{s}}^{M_\mathrm{s}} \big(\mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_\mathrm{s}}\big)(\boldsymbol{x}) \right).\end{aligned} \tag{3.72}$$

We see that the sum over m_o halves, so the expected speed gain is approximately a factor 2.

3.7.3 Orientation Scores Obtained by Lifting

In some methods, an orientation score is obtained by so-called *lifting*. Suppose that we have at every image position $\boldsymbol{x} \in \mathbb{R}^2$ a single hypothetical orientation

$\beta(\boldsymbol{x})$ and corresponding feature strength $s(\boldsymbol{x})$. In this case we can consider this feature as a delta spike in θ and lift this to the appropriate orientation,

$$U(\boldsymbol{x},\theta) = s(\boldsymbol{x})\delta(\theta - \beta(\boldsymbol{x})), \tag{3.73}$$

where δ is the Dirac delta function. Substituting the latter equation in (3.17) gives

$$(\Psi *_{SE(2)} U)(\boldsymbol{x},\theta) = \int_{\mathbb{R}^2} \Psi(\mathbf{R}^{-1}_{\beta(\boldsymbol{x}')}(\boldsymbol{x}-\boldsymbol{x}'), \theta - \beta(\boldsymbol{x}'))s(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'. \tag{3.74}$$

The integral over θ has disappeared, leaving an integral over the spatial dimensions only. For a direct implementation, the complexity becomes $\mathcal{O}\left(N_o N_s^2 K_s^2\right)$. A non-steerable FFT implementation is not feasible in this case, since a lifted orientation score cannot be sampled in the orientation dimension.

The orientation-angular Fourier decomposition of a lifted orientation score is given by $\mathcal{F}_\theta[U]_{m_\mathrm{o}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}} e^{-im_\mathrm{o}\beta(\boldsymbol{x})} s(\boldsymbol{x})$. The steerable $SE(2)$-convolution in this case becomes, by substituting the latter equation in (3.49)

$$\mathcal{F}_\theta[\Psi *_{SE(2)} U]_{m_\mathrm{o}}(\boldsymbol{x}) = \sum_{m_\mathrm{s}\in\mathbb{Z}} \mathcal{C}[\Psi]_{m_\mathrm{o},m_\mathrm{s}} * (s\, e^{-im_\mathrm{s}\beta}). \tag{3.75}$$

Note that a lifted orientation score is not orientation-angularly bandlimited, meaning that the $SE(2)$-kernel should be spatial-angularly bandlimited to ensure the sum over m_s to be finite.

The complexity coincides with the complexity of the steerable $SE(2)$-convolution using FFT, viz. $\mathcal{O}\left(N_s^2(N_o \log N_s + C_o C_s)\right)$.

3.7.4 Separable $SE(2)$-Kernels

A separable $SE(2)$-kernel is defined as a kernel that can be written as

$$\Psi(\boldsymbol{x},\theta) = \Psi_\mathrm{S}(\mathbf{R}_\theta^{-1}\boldsymbol{x})\Psi_\mathrm{O}(\theta). \tag{3.76}$$

Substituting the latter equation in (3.17) gives

$$\begin{aligned}
&(\Psi *_{SE(2)} U)(\boldsymbol{x},\theta) \\
&\quad = \int_{\mathbb{R}^2}\int_0^{2\pi} \Psi_\mathrm{S}(\mathbf{R}^{-1}_{\theta-\theta'}\mathbf{R}^{-1}_{\theta'}(\boldsymbol{x}-\boldsymbol{x}'))\Psi_\mathrm{O}(\theta-\theta')U(\boldsymbol{x}',\theta')\mathrm{d}\theta'\mathrm{d}\boldsymbol{x}' \\
&\quad = \int_{\mathbb{R}^2} \Psi_\mathrm{S}(\mathbf{R}_\theta^{-1}(\boldsymbol{x}-\boldsymbol{x}')) \left(\int_0^{2\pi} \Psi_\mathrm{O}(\theta-\theta')U(\boldsymbol{x}',\theta')\mathrm{d}\theta' \right) \mathrm{d}\boldsymbol{x}'.
\end{aligned} \tag{3.77}$$

The $SE(2)$-convolution can be calculated by first applying 1-dimensional convolutions with Ψ_{O} in the orientation dimension, followed by 2-dimensional convolutions with $\Psi_{\mathrm{S}}^{\theta}$ in the spatial dimensions. This is similar to a separable $\mathbb{R}^2$-convolution, but in this case the two separated convolution operations do *not commute*. The complexity in this case is $\mathcal{O}\left(N_o N_s^2 \log N_s\right)$ with FFT and $\mathcal{O}\left(N_o N_s^2 K_s^2\right)$ without FFT.

We can also make the spatial part Ψ_{S} steerable in the "classical sense" as described in Subsection 3.4.1. However, this is not beneficial since in this case the complexity multiplies with a factor C_s.

3.8 Stochastic Completion Fields

In Subsection 2.9.2.1 we introduced stochastic completion kernels and introduced the operational definition for calculating stochastic completion fields. In literature, many numerical approaches can be found to calculate stochastic completion fields, e.g. the iterative numerical scheme by Zweck and Williams [155] and the algorithm by August [7]. In this section we will only consider the calculation of stochastic completion fields using $SE(2)$-convolutions. The advantages of this approach are: the potential accuracy given that the sampled Green's function of the completion process is accurately sampled; the fact that the $SE(2)$-convolution is composed of standard operations, namely FFT, convolution, matrix multiplication, that can be optimized substantially for instance by parallelization or implementation on an FPGA or a GPU; and the versatility, i.e. for all linear left-invariant operations we can use the same algorithm.

Van Almsick and Duits [3] [35] derive several analytic expressions for the stochastic completion kernel. Several expressions for the *exact* solution in Fourier space are provided by these authors, from which one can derive analytic expressions for the steerable components. However, the evaluation of these expressions turns out the be computationally rather demanding if the analytic solutions are sampled, since it involves sampling of Mathieu functions. Alternatively, the August algorithm [7] can be regarded as the numerical equivalent to the analytic derivations of the Greens functions in [35]. However, this algorithm is demanding as well as it involves numerical band-diagonal $N_o \times N_o$ matrix inversion for each spatial pixel position. However, van Almsick and Duits propose several approximations that turn out to suffice in practice. In this section we restrict to the polar approximation, which is given by

$$K(r,\phi,\theta) = \frac{\sqrt{3}}{\pi(\sigma r)^2} e^{-\lambda r - \frac{(2\theta-3\phi)^2+3\phi^2}{2\sigma^2 r}}. \tag{3.78}$$

Figure 3.4 and Figure 3.5 show examples, of stochastic completion fields of arti-

Figure 3.4: Example stochastic completion field for contour completion on the TU/e logo. (a) Input image. (b) Result.

ficial images, taken from [27]. These results are obtained using by calculating an orientation score using the transformation described in Subsection 2.4.1, where only the real-valued part is maintained. This orientation score is squared, i.e. $\mathrm{Re}(U_f)^2$, which has the effect of enhancing the contrast. Then the completion field is calculated, cf. (2.97) on page 48, using the non-steerable $SE(2)$-convolution implementation as described in Subsection 3.3.3. Finally, a new image is obtained by the inverse orientation score transformation. Clearly, a stochastic completion field tends to close gaps in line structures. However, it also leads to blurring orthogonal to the structures of interest.

The completion kernel (3.78) is not angularly bandlimited. To apply an $SE(2)$-convolution on a sampled orientation score while adhering to the bandwidth requirement of Subsection 3.5.3 one can use a steerable $SE(2)$-convolution and truncate the orientation-angular and spatial-angular frequency components. We can determine the steerable components for this stochastic completion kernel using (3.44), yielding

$$\mathcal{C}[K]_{m_\mathrm{o},m_\mathrm{s}}(r,\phi) = \frac{1}{r\sqrt{2\pi}} e^{-r\lambda - \frac{1}{6}(7{m_\mathrm{o}}^2 - 5m_\mathrm{o},m_\mathrm{s} + {m_\mathrm{s}}^2)r\sigma^2} e^{i(m_\mathrm{s}-m_\mathrm{o})\phi}. \tag{3.79}$$

The $\mathbb{R}^2$-Fourier transform of these steerable components are given by

$$\mathcal{F}_{\mathbb{R}^2}[\mathcal{C}[K]_{m_\mathrm{o},m_\mathrm{s}}](\boldsymbol{\omega}) = \begin{cases} \frac{1}{\sqrt{2\pi}\sqrt{d^2+\omega_x^2+\omega_y^2}} \left(\frac{-\omega_y + i\,\omega_x}{d+\sqrt{d^2+\omega_x^2+\omega_y^2}} \right)^{m_\mathrm{s}} & \text{if } m_\mathrm{s} > 0; \\ \frac{1}{\sqrt{2\pi}\sqrt{d^2+\omega_x^2+\omega_y^2}} & \text{if } m_\mathrm{s} = 0; \\ -\frac{1}{\sqrt{2\pi}\sqrt{d^2+\omega_x^2+\omega_y^2}} \left(\frac{\omega_y - i\,\omega_x}{d+\sqrt{d^2+\omega_x^2+\omega_y^2}} \right)^{-m_\mathrm{s}} & \text{if } m_\mathrm{s} < 0, \end{cases}$$
$$\text{with } d = \lambda + \frac{1}{6}(7{m_\mathrm{o}}^2 - 5m_\mathrm{o}, m_\mathrm{s} + {m_\mathrm{s}}^2)\sigma^2. \tag{3.80}$$

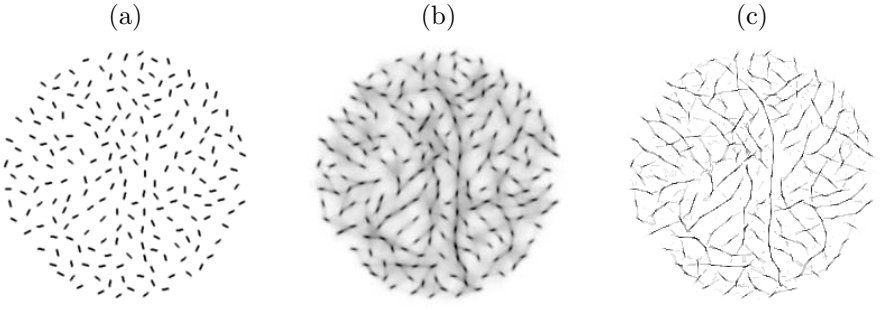


Figure 3.5: Example stochastic completion field for contour completion on a sparse set of contour segments. (a) Input image. (b) Resulting stochastic completion field. (c) Result after thinning, extracting the centerlines of the contours.

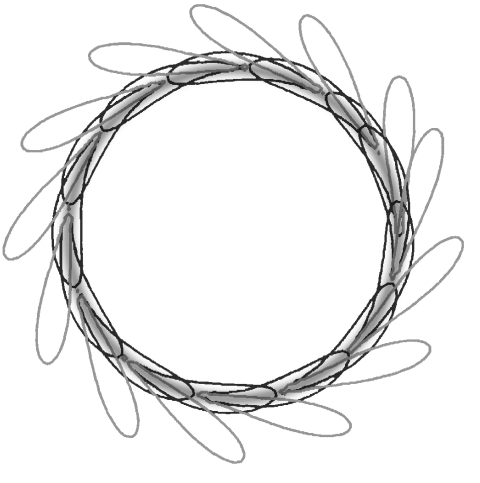


Figure 3.6: Illustration of the difference between stochastic completion kernels with and without curvature. The kernels without curvature lead to more spurious responses perpendicular to the line structure, while the kernels with curvature fits better to the curved line.

3.8.1 Curvature-Adaptive Stochastic Completion Fields

For the non-steerable direct implementation of the $SE(2)$-convolution it is a straightforward extension of the algorithm to use a different kernel at each position $(\boldsymbol{x}, \theta)$. In [73], we have investigated the possibility to make the stochastic completion field curvature-adaptive. The advantage of this approach is illustrated is Figure 3.6. A curved stochastic completion kernel with a curvature drift κ can be obtained by using the following parameters in equation (2.90)

$$\boldsymbol{a} = (1, 0, \kappa), \quad \mathbf{D} = \mathrm{diag}(0, 0, \sigma^2). \tag{3.81}$$

A curvature-adaptive $SE(2)$-convolution is defined by

$$(\Psi_\kappa *^{\mathrm{adaptive}}_{SE(2)} U)(\boldsymbol{x}, \theta) = \int_{\mathbb{R}^2} \int_0^{2\pi} \Psi_{\kappa(\boldsymbol{x})}(\mathbf{R}_{\theta'}^{-1}(\boldsymbol{x} - \boldsymbol{x}'), \theta - \theta') U(\boldsymbol{x}', \theta') \, \mathrm{d}\theta' \, \mathrm{d}\boldsymbol{x}'. \tag{3.82}$$

where Ψ_κ denotes a completion kernel with curvature κ, where the fixed parameters σ and λ are omitted for notational convenience. The curvature adaptive completion field is obtained by mean of a product of a forward and a backward adaptive $SE(2)$-convolution. The required curvature κ can be estimated with the method that will be described in Chapter 5. Figure 3.7, taken from [73], shows some results obtained with this approach.

3.9 Orientation Channel Smoothing

In this section we show that orientation channel smoothing [105, 60, 45] is another method that can easily be described in terms of $SE(2)$-convolutions. For each spatial position $\boldsymbol{x}$, the input to the orientation channel smoothing method is a single hypothetical orientation $\beta(\boldsymbol{x})$ and a measure $s(\boldsymbol{x})$ for the confidence of having an oriented structure with that orientation. s and β are encoded in a so-called channel representation, defined by

$$U_{\mathrm{CR}}(\boldsymbol{x}, l) = s(\boldsymbol{x}) B(\beta(\boldsymbol{x}) - l \cdot s_\theta), \tag{3.83}$$

where $l \in \{0, 1, \ldots, N_o - 1\}$, N_o is the number of samples in the orientation dimension, and $s_\theta = p/N_o$ where $p = \pi$ or 2π depending on the periodicity of the orientation score. B denotes the *encoding function* that is used, often a B-spline or a $\cos^2$-function. This encoding function should be chosen such that it is possible for each channel column $l \mapsto U_{\mathrm{CR}}(\boldsymbol{x}, l)$ to reconstruct the continuous input values of $s(\boldsymbol{x})$ and $\beta(\boldsymbol{x})$ out of the sampled channel. This process is illustrated in Figure 3.8a.

The channel representation U_{CR} is smoothed separately for each orientation layer $U_{\mathrm{CR}}(\cdot, l)$ by means of an $\mathbb{R}^2$-convolution with a rotating anisotropic kernel such

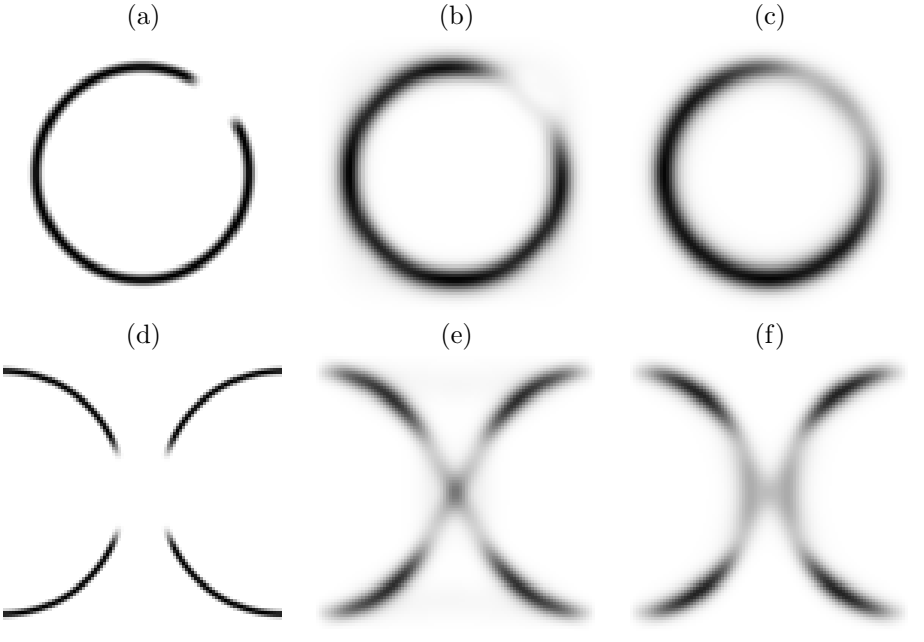


Figure 3.7: Example of curvature-adaptive stochastic completion field. (a)+(d) input images. (b)+(e) stochastic completion field without curvature. (c)+(f) curvature-adaptive stochastic completion field.

as [37]

$$\Psi_{\mathrm{S}}(\boldsymbol{x}) = \begin{cases} e^{-r\,\varsigma}\cos^2\varpi\phi & \text{if } |\varpi\phi| \leq \pi/2, \\ 0 & \text{otherwise,} \end{cases} \tag{3.84}$$

where ς is a parameter determining the spatial extent of the kernel and ϖ determines the "opening angle" of the kernel. The effective operational definition is

$$W_{\mathrm{CR}}(\boldsymbol{x}, l) = \int_{\mathbb{R}^2} \Psi_{\mathrm{S}}(\mathbf{R}_\theta^{-1}(\boldsymbol{x} - \boldsymbol{x}'))s(\boldsymbol{x}')B(\beta(\boldsymbol{x}') - l\,s_\theta)\mathrm{d}\boldsymbol{x}', \tag{3.85}$$

The goal is to get enhanced (robust) values for the features s and β from W_{CR}. To be insensitive to outliers, these values are obtained by determining the position and strength of the *mode* at each spatial position, i.e.

$$s(\boldsymbol{x}) = \max_\theta W(\boldsymbol{x}, \theta), \quad \beta(\boldsymbol{x}) = \arg\max_\theta W(\boldsymbol{X}, \theta) \tag{3.86}$$

The channel method uses a computationally efficient decoding scheme called *windowed decoding* to approximate these modes. This is illustrated in Figure 3.8b-c: a window of a fixed length is selected in the channel that contains the largest signal strength, and the reconstruction of the value is performed using only values on positions inside of this window. The drawback of this method is that it suffers from quantization artefacts. Therefore, in [45] it is proposed to determine the mode more accurately using Fourier series interpolation.

Now, we come to the observation that channel smoothing can be described in terms of a separable $SE(2)$-convolution as described in Subsection 3.7.4. By choosing

$$\Psi(\boldsymbol{x}, \theta) = \Psi_{\mathrm{S}}(\mathbf{R}_\theta^{-1}\boldsymbol{x})B(\theta), \qquad U_{\mathrm{CR}}(\boldsymbol{x}, \theta) = s(\boldsymbol{x})\delta(\theta - \beta(\boldsymbol{x})), \tag{3.87}$$

the channel smoothing operation cf. equation (3.85) equals $W = \Psi *_{SE(2)} U$.

3.10 Conclusions

We discussed non-steerable and steerable $SE(2)$-convolutions and their implementations. To give an overview, Table 3.2 on page 56 summarizes the algorithmic complexities of the different implementations of $SE(2)$-convolutions that have been mentioned.

In general, steerable implementations can be faster if the total number of steerable components, $C_s \cdot C_o$, is lower than the product of the number of orientations of the orientation score and the number of effective orientations of the $SE(2)$-kernel $N_o \cdot K_o$. Besides the advantage that no interpolations are needed, it is also advantageous than one has full control over the Gibbs oscillations and aliasing

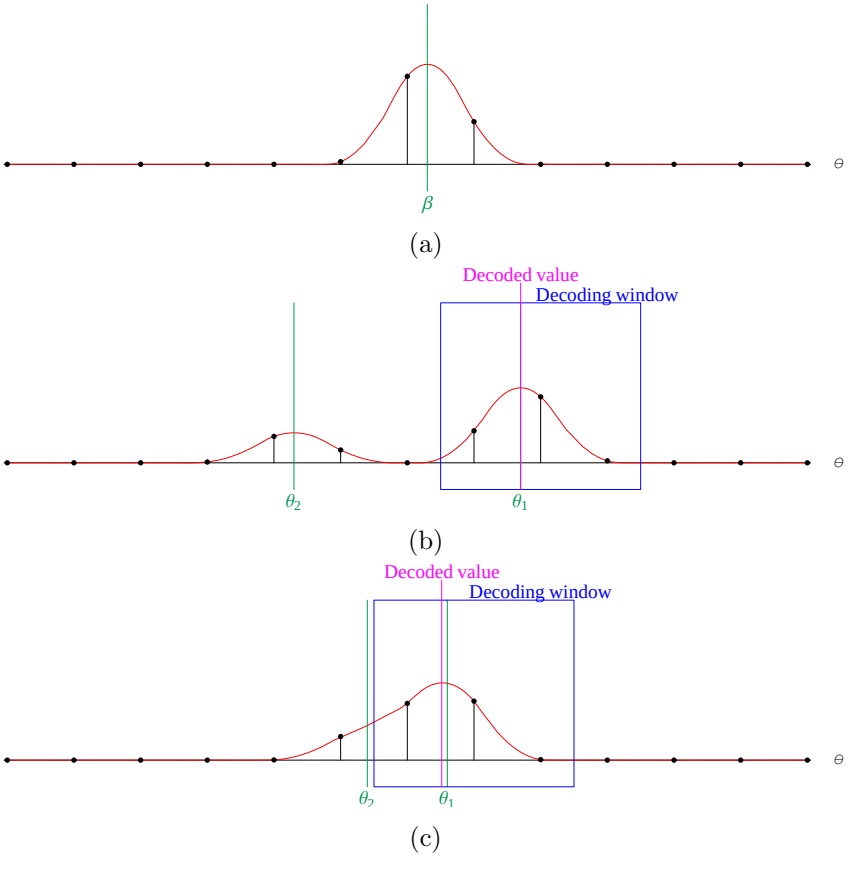


Figure 3.8: (a) Example of channel encoding of an orientation $\theta = \beta$. Although the input orientation value β is not aligned with a sample point (indicated by the black dots), it is possible to reconstruct the value of β out of the sample values, if one knows the encoding function B. (b) Windowed decoding: reconstruction is performed within a window of fixed size, which is placed such that signal strength inside the window is maximized. In this example, two channels with encoded orientations θ_1 and θ_2 were added. Because θ_1 is stronger, this value is decoded, while θ_2 is regarded as outlier. (c) Example of windowed decoding, where θ_1 and θ_2 are interfering, causing the decoded value to lie in-between both orientations.

artefacts that can occur if the sampling theorem on orientation scores as described in Subsection 3.5.3.2 is not fulfilled. Furthermore, we showed that our steerable approach leads to reduction in the number of computations and a reduction in complexity compared to the $SE(2)$-convolution implementation as proposed by Chirkjian [22].

Subsequently, we addressed a number of common simplifying special cases. We also showed that stochastic completion fields and channel representations fit in the framework of orientation scores processing. In the next chapter we will show that orientation scores provide a framework for tensor-valued image processing as well, and that steerable $SE(2)$-convolution is also applicable in that case.

Acknowledgements

Parts of the work on steerable $SE(2)$-convolutions has been performed in collaboration with Markus van Almsick (Technische Universiteit Eindhoven). Renske de Boer and Gijs Huisman have contributed to Section 3.8 during their final master projects.

Profound study of nature is the most fertile source of mathematical discoveries.

Jean Baptiste Joseph Fourier (1768-1830)

4 Tensor-Valued Images and Tensor Voting

This chapter is partly based on:

E. M. Franken, M.A. van Almsick, P. Rongen, L. M. J. Florack, and B. M. ter Haar Romeny. An efficient method for tensor voting using steerable filters. In *Proceedings of the Ninth European Conference on Computer Vision (ECCV)*, volume 3954 of LNCS, pages 228–240, 2006.

E. M. Franken, P. Rongen, M. van Almsick, and B. M. ter Haar Romeny. Detection of electrophysiology catheters in noisy fluoroscopy images. In *Proceedings of the 9th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI) 2006*, volume 4191 of LNCS, pages 25–32, 2006.

4.1 Introduction

There is a one-to-one relation between tensor images and orientation scores, meaning that tensor images can be considered as functions on $SE(2)$ as well, as we will show in this chapter. Using this relation, we can in principle apply all orientation score methods described in this thesis to tensor images as well. Many existing tensor-based methods can be expressed as $SE(2)$-convolutions, giving better insights in how these methods exactly work. Especially of interest in this context is tensor voting [64, 98]. We will show that *steerable tensor voting* leads to a complexity reduction due to the fact that the fast Fourier transform can be used. Subsequently we also put structure tensor approaches [50, 86] into the orientation score framework. Finally, we will apply steerable tensor voting to a medical image analysis problem, namely electrophysiology (EP) catheter extraction.

4.2 Convolutions on 2-Tensor Fields

In this section we show how tensor images relate to orientation scores and how the principles of steerable $SE(2)$-convolution can be directly mapped to this case. We will propose a convolution operation on tensor-valued images, which, in contrast to e.g. the tensor-convolution described in [71], adheres to the $SE(2)$ group structure.

4.2.1 Basic Definition of 2-Tensors

In mathematics, a tensor is defined in an abstract way as a multilinear function that takes a number of vectors / covectors on its arguments and yields a scalar value. A tensor with n inputs is called an n-tensor. Once a basis is defined, each 1-tensor can be identified with a vector, each 2-tensor can be identified with a matrix, and higher order tensors can be identified with higher-dimensional entities, i.e. holors.

We only consider *two-dimensional symmetric 2-tensors*, since this is the type of tensor that normally occurs in image processing, e.g. the structure tensor, the Hessian matrix, tensor voting, and DTI (Diffusion Tensor Imaging) tensors. A two-dimensional symmetric 2-tensor $a \in \mathbf{T}_2(\mathbb{R}^2)$ is a function $\mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ given by

$$a(\boldsymbol{v}, \boldsymbol{w}) = \boldsymbol{v}^{\mathrm{T}} \cdot \mathbf{A} \cdot \boldsymbol{w}, \quad \text{with } a(\boldsymbol{v}, \boldsymbol{w}) = a(\boldsymbol{w}, \boldsymbol{v}), \quad \text{and } \mathbf{A} = \begin{pmatrix} a_{xx} & a_{xy} \\ a_{xy} & a_{yy} \end{pmatrix}, \tag{4.1}$$

where $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^2$ are vectors where the vector components are expressed with respect to the standard Cartesian coordinate system $\{\boldsymbol{e}_x, \boldsymbol{e}_y\}$. Since symmetric matrix $\mathbf{A}$ fully describes tensor a we will further on also refer to $\mathbf{A}$ as being a tensor, as is common practice in most tensor image processing literature.

4.2.2 Rotation of 2-Tensors

A rotation of a 2-tensor is achieved by counter-rotating the input vectors of the tensor, i.e.

$$\mathcal{R}_\alpha[a](\boldsymbol{v}, \boldsymbol{w}) = a(\mathbf{R}_\alpha^{-1}\boldsymbol{v}, \mathbf{R}_\alpha^{-1}\boldsymbol{w}) = \boldsymbol{v}^{\mathrm{T}}(\mathbf{R}_\alpha \mathbf{A}\, \mathbf{R}_\alpha^{-1})\boldsymbol{w}, \tag{4.2}$$

in which we have used the property $\mathbf{R}_\alpha^{-1} = \mathbf{R}_\alpha^{\mathrm{T}}$. Rotation of the matrix $\mathbf{A}$ is given by

$$\mathcal{R}_\alpha[\mathbf{A}] = \mathbf{R}_\alpha \mathbf{A}\, \mathbf{R}_\alpha^{-1}. \tag{4.3}$$

If we collect the components a_{xx}, a_{xy}, and a_{yy} in a column vector and do some trigonometric calculations, we can write the rotation of $\mathbf{A}$ as[1]

$$\mathcal{R}_\alpha\left[\begin{pmatrix} a_{xx} \\ a_{xy} \\ a_{yy} \end{pmatrix}\right] = \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2\pi}} & \frac{1}{\sqrt{2\pi}} & \frac{1}{\sqrt{2\pi}} \\ -\frac{i}{\sqrt{2\pi}} & 0 & \frac{i}{\sqrt{2\pi}} \\ -\frac{1}{\sqrt{2\pi}} & \frac{1}{\sqrt{2\pi}} & -\frac{1}{\sqrt{2\pi}} \end{pmatrix}}_{\mathbf{S}^{-1}} \underbrace{\begin{pmatrix} e^{2i\,\alpha} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & e^{-2i\,\alpha} \end{pmatrix}}_{\mathbf{R}'_\alpha} \underbrace{\begin{pmatrix} \frac{1}{2}\sqrt{\frac{\pi}{2}} & i\sqrt{\frac{\pi}{2}} & -\frac{1}{2}\sqrt{\frac{\pi}{2}} \\ \sqrt{\frac{\pi}{2}} & 0 & \sqrt{\frac{\pi}{2}} \\ \frac{1}{2}\sqrt{\frac{\pi}{2}} & -i\sqrt{\frac{\pi}{2}} & -\frac{1}{2}\sqrt{\frac{\pi}{2}} \end{pmatrix}}_{\mathbf{S}} \begin{pmatrix} a_{xx} \\ a_{xy} \\ a_{yy} \end{pmatrix}, \tag{4.4}$$

so in correspondence to Subsection 3.4.2, we are capable of diagonalizing the rotation matrix. The irreducible representations for rotation of 2-tensors are $e^{-im_o\alpha}$, $m_o \in \{-2, 0, 2\}$, and in the new basis the components are given by

$$\begin{pmatrix} \hat{A}_{-2} \\ \hat{A}_0 \\ \hat{A}_2 \end{pmatrix} = \mathbf{S} \cdot \begin{pmatrix} a_{xx} \\ a_{xy} \\ a_{yy} \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\sqrt{\frac{\pi}{2}}(a_{xx} - a_{yy} + 2i\, a_{xy}) \\ \sqrt{\frac{\pi}{2}}(a_{xx} + a_{yy}) \\ \frac{1}{2}\sqrt{\frac{\pi}{2}}(a_{xx} - a_{yy} - 2i\, a_{xy}) \end{pmatrix} \tag{4.5}$$

The same transform can also be defined directly on the 2×2 matrices

$$\hat{A}_{m_o} = (\mathbf{A}, \mathbf{E}_{m_o}) = \mathrm{tr}(\mathbf{A}, \overline{\mathbf{E}_{m_o}}^{\mathrm{T}}), \quad m_o \in \{-2, 0, 2\}, \tag{4.6}$$

where $\mathbf{E}_{m_o}$ is an orthogonal basis with respect to the inner product $(\mathbf{A}, \mathbf{B}) = \mathrm{tr}(\mathbf{A}, \overline{\mathbf{B}}^{\mathrm{T}})$, given by

$$\mathbf{E}_{-2} = \frac{1}{2}\sqrt{\frac{\pi}{2}}\begin{pmatrix} 1 & -i \\ -i & -1 \end{pmatrix} \qquad \mathbf{E}_0 = \sqrt{\frac{\pi}{2}}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{E}_2 = \frac{1}{2}\sqrt{\frac{\pi}{2}}\begin{pmatrix} 1 & i \\ i & -1 \end{pmatrix}. \tag{4.7}$$

The inverse transform is given by

$$\mathbf{A} = \hat{A}_{-2}\hat{\mathbf{E}}_{-2} + \hat{A}_0\hat{\mathbf{E}}_0 + \hat{A}_2\hat{\mathbf{E}}_2, \tag{4.8}$$

where

$$\hat{\mathbf{E}}_{m_o} = \frac{1}{\pi}(2 - \delta_{m_o})\mathbf{E}_{m_o}. \tag{4.9}$$

[1] Here the normalization of $\mathbf{S}$ is chosen such that later on in this subsection it is consistent to the angular Fourier decomposition on a function $\mathbb{T} \to \mathbb{C}$ cf. (3.5) on page 51 that we construct from $\mathbf{A}$ using (4.10)

In conclusion, a tensor of this type has exactly the same rotational properties as a function $A : \mathbb{T} \to \mathbb{R}$ with nonzero angular frequency components $\mathcal{F}_{\mathbb{T}}[A]_{m_o}$ for $m_o \in \{-2, 0, 2\}$. The relation between a 2-tensor $\mathbf{A}$ and a function on $\mathbb{T}$ is given by the following important direct formula

$$A(\theta) = \boldsymbol{n}(\theta)^{\mathrm{T}} \mathbf{A} \boldsymbol{n}(\theta), \quad \text{with } \boldsymbol{n}(\theta) = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \tag{4.10}$$

where $\mathcal{F}_{\mathbb{T}}[A]_{m_o} = \hat{A}_{m_o} = (\mathbf{A}, \mathbf{E}_{m_o})$.

4.2.3 The Convolution

From the relation between 2-tensors and functions $\mathbb{T} \to \mathbb{C}$ described above, it automatically follows that a tensor-valued image is isomorphic to an orientation score with orientation-angular frequency components $m_o \in \{-2, 0, 2\}$. Consequently, the appropriate way to perform a tensor-convolution, more specific a convolution of a 2-tensor-valued image with a 2-tensor-valued kernel, follows directly from equation (3.49) on page 64

$$(\mathbf{V} * \mathbf{U})(\boldsymbol{x}) = \sqrt{2\pi} \sum_{m_o \in \{-2,0,2\}} \sum_{m_s \in \{-2,0,2\}} \big(\hat{V}_{m_o,m_s} *_{\mathbb{R}^2} \hat{U}_{m_s}\big)(\boldsymbol{x}) \hat{\mathbf{E}}_{m_o}, \tag{4.11}$$

where

$$\hat{U}_{m_s}(\boldsymbol{x}) = (\mathbf{U}(\boldsymbol{x}), \mathbf{E}_{m_s}), \quad \text{and} \tag{4.12}$$

$$\hat{V}_{m_o,m_s}(\boldsymbol{x}) = \mathcal{C}[(\boldsymbol{x}, \theta) \mapsto \boldsymbol{n}(\theta)^{\mathrm{T}} \mathbf{V}(\boldsymbol{x}) \boldsymbol{n}(\theta)]_{m_o,m_s}(\boldsymbol{x}). \tag{4.13}$$

To fulfill the sampling theorem in Subsection 3.5.3.2, the inherent orientation-angular bandlimit ($m_o \in \{-2, 0, 2\}$) of tensor-valued image $\mathbf{U}$ restricts the admissible kernels $\mathbf{V}$ to steerable tensor-valued filters with spatial-angular bandlimit $m_s \in \{-2, 0, 2\}$.

4.2.4 Semi-Positive Definite Tensors

In all 2-tensor-valued image processing methods mentioned in Subsection 4.2.1, except for the Hessian matrix, only *semi-positive definite* symmetric 2-tensors are possible, i.e. tensors with nonnegative eigenvalues. In that case a relevant question is whether this property is preserved after convolution with some kernel $\mathbf{V}$. To this end, note that the eigenvalues $\lambda_1 \geq \lambda_2 \geq 0$ can be easily described in terms of $\hat{A}_{m_o}$.

$$\begin{aligned} \lambda_1 &= \frac{1}{\sqrt{2\pi}} \left(\hat{A}_0 + 2\sqrt{A_{-2} A_2} \right) = \frac{1}{\sqrt{2\pi}} \left(\hat{A}_0 + 2|A_2| \right), \\ \lambda_2 &= \frac{1}{\sqrt{2\pi}} \left(\hat{A}_0 - 2|A_2| \right), \end{aligned} \tag{4.14}$$

Furthermore, since

$$\begin{aligned} A(\theta) &= \frac{1}{\sqrt{2\pi}}\left(\hat{A}_0 + \hat{A}_{-2}e^{-2i\theta} + \hat{A}_2 e^{2i\theta}\right) \\ &= \frac{1}{\sqrt{2\pi}}\left(\hat{A}_0 + (\hat{A}_2 + \hat{A}_{-2})\cos 2\theta + (\hat{A}_2 - \hat{A}_{-2})\sin 2\theta\right) \end{aligned} \tag{4.15}$$

we observe that the range of A is given by $\lambda_2 \leq A(\theta) \leq \lambda_1$ for all θ. Consequently, if $\mathbf{A}$ is semi-positive definite, then $A(\theta)$ is nonnegative for all θ, so a semi-positive definite tensor-valued image corresponds to a nonnegative orientation score. For $SE(2)$-convolutions on orientation scores it is straightforward that nonnegativity of an orientation score is always preserved iff the $SE(2)$-kernel is nonnegative. From this it follows that in the case of 2-tensor convolution, semi-positive definiteness is always preserved if $\mathbf{V}$ is a semi-positive definite tensor-valued kernel.

4.3 Tensor Voting

Tensor voting [64, 98, 99] is a technique for grouping and extraction of lines and contours. The basic idea is that this process is made more robust by applying a orientation-dependent smoothing operation on a tensor field. In this section we will discuss tensor voting in more detail, show how it fits in our framework, and propose a *steerable* tensor voting method.

4.3.1 The Tensor Voting Operation

The input and output of tensor voting are tensor fields, which are denoted by $\mathbf{U}$ and $\mathbf{W}$ respectively. The method does not prescribe how to obtain the input tensor field $\mathbf{U}$. It is considered an application-specific free choice. Medioni et al. [98] assume that the input tensor field $\mathbf{U}$ is sparse[2], i.e. that most of the tensors in $\mathbf{U}$ are zero. Here we do not make any assumption about sparseness; the input field $\mathbf{U}$ may be either dense or sparse.

Tensor voting uses the following three properties to uniquely describe and interpret a tensor $\mathbf{A}$,

$$\text{Orientation} \quad \beta(\mathbf{A}) = \arccos\big(\boldsymbol{e}_1(\mathbf{A}) \cdot \boldsymbol{e}_x\big) \quad = \frac{1}{2}\arg(a_{xx} - a_{yy} + 2i\,a_{xy}), \tag{4.16}$$

$$\text{Stickness} \quad s(\mathbf{A}) = \lambda_1(\mathbf{A}) - \lambda_2(\mathbf{A}) = \sqrt{\mathrm{tr}(\mathbf{A})^2 - 4\,\det\mathbf{A}}, \tag{4.17}$$

$$\text{Ballness} \quad b(\mathbf{A}) = \lambda_2(\mathbf{A}) = \frac{1}{2}\left(\mathrm{tr}(\mathbf{A}) - \sqrt{\mathrm{tr}(\mathbf{A})^2 - 4\,\det\mathbf{A}}\right), \tag{4.18}$$

[2] Medioni et al. use a different terminology and call this a sparse set of tokens.

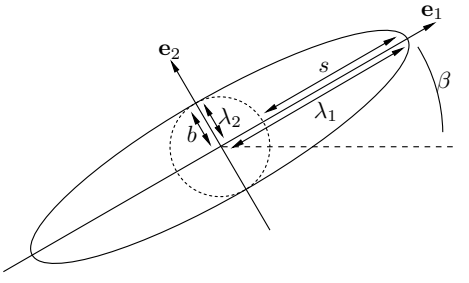


Figure 4.1: Graphical representation of a symmetric semi-positive definite 2-tensor.

where $\lambda_1(\mathbf{A}) \geq \lambda_2(\mathbf{A}) \geq 0$ are the eigenvalues of $\mathbf{A}$ and $\boldsymbol{e}_1(\mathbf{A})$ is the eigenvector corresponding to the largest eigenvalue. The stickness is a measure for *orientation certainty* or anisotropy. The ballness is a measure for *orientation uncertainty* or isotropy. For an intuitive graphical illustration, we can identify tensor $\mathbf{A}$ with an ellipse $\mathcal{E}_{\mathbf{A}}$ defined as [3]

$$\mathcal{E}_{\mathbf{A}} = \left\{ \boldsymbol{x} | \boldsymbol{x}^{\mathrm{T}} \mathbf{A}^{-2} \boldsymbol{x} = 1 \right\}. \tag{4.19}$$

The eigenvalues of $\mathbf{A}$ determine the radii of the ellipse, while the eigenvectors determine the principal axes. An example of such an ellipse is shown Figure 4.1.

The operational definition of tensor voting is given by

$$\begin{aligned} \mathbf{W}(\boldsymbol{x}) = &\int_{\mathbb{R}^2} \mathcal{R}_{\beta(\mathbf{U}(\boldsymbol{x}'))}[\mathbf{V}](\boldsymbol{x} - \boldsymbol{x}')\, s(\mathbf{U}(\boldsymbol{x}'))\, \mathrm{d}\boldsymbol{x}' \\ &+ \int_{\mathbb{R}^2} b(\mathbf{U}(\boldsymbol{x}'))\, \mathbf{V}_{\mathrm{B}}(\boldsymbol{x} - \boldsymbol{x}')\, \mathrm{d}\boldsymbol{x}', \end{aligned} \tag{4.20}$$

where $\mathcal{R}_{\beta(\mathbf{U}(\boldsymbol{x}'))}[\mathbf{V}]$ is an anisotropic tensor-valued kernel called *stick voting field* that is rotated over $\beta(\mathbf{U}(\boldsymbol{x}'))$ by

$$\mathcal{R}_{\alpha}[\mathbf{V}](\boldsymbol{x}) = \mathbf{R}_{\alpha} \mathbf{V}(\mathbf{R}_{\alpha}^{-1} \boldsymbol{x}) \mathbf{R}_{\alpha}^{-1}, \tag{4.21}$$

and $\mathbf{V}_{\mathrm{B}}$ is a rotationally symmetric tensor-valued kernel called *ball voting field*. The first integral enables the process that is called *stick voting*, while the second integral enables the process called *ball voting*. The complexity of a direct implementation of this operation is $\mathcal{O}\left(N_s^2 K_s^2\right)$.

The intuition behind the tensor voting operation is to let tensors *communicate* with each other by adding up contributions of neighboring tensors, resulting in

[3] This equation is obtained as follows. The general equation for an ellipse is $\boldsymbol{x}^{\mathrm{T}} \mathbf{M} \boldsymbol{x} = 1$ with positive-definite matrix $\mathbf{M}$. We know that $\boldsymbol{e}_i(\mathbf{A})^{\mathrm{T}} \mathbf{A}^n \boldsymbol{e}_i(\mathbf{A}) = \lambda_i(\mathbf{A})^n$ where the eigenvectors have unit length. To get the ellipse as illustrated in Figure 4.1 we have to ensure that $\left(\lambda_i(\mathbf{A})\boldsymbol{e}_i(\mathbf{A})\right)^{\mathrm{T}} \mathbf{M} \left(\lambda_i(\mathbf{A})\boldsymbol{e}_i(\mathbf{A})\right) = 1$ for $i \in \{1, 2\}$, which is accomplished by setting $\mathbf{M} = \mathbf{A}^{-2}$.

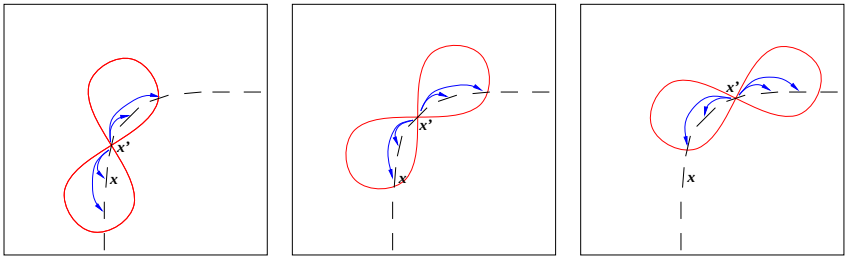


Figure 4.2: Illustration of the tensor voting operation: the tensors communicate with each other using the stick voting field, the essentially compact support of which is indicated by the "8-shaped" contour. In this way the tensors strengthen each other. In the left figure, $\boldsymbol{x}$ receives a strong contribution of $\boldsymbol{x}'$, in the middle figure the contribution of $\boldsymbol{x}'$ is weak due to the larger distance, while in the right figure $\boldsymbol{x}$ does not get any significant contribution from $\boldsymbol{x}'$

a context-enhanced tensor field $\mathbf{W}$. Figure 4.2 illustrates the way the stick voting field is used on input data consisting of stick tensors. For each (nonzero) tensor $\mathbf{U}(\boldsymbol{x}')$, the voting field $\mathcal{R}_\alpha[\mathbf{V}]$ is centered at position $\boldsymbol{x}'$, aligned with the local orientation $\beta(\mathbf{U}(\boldsymbol{x}'))$: $\mathcal{R}_{\beta(\mathbf{U}(\boldsymbol{x}'))}[\mathbf{V}](\boldsymbol{x}-\boldsymbol{x}')$. Then, a weighted contribution (called a *vote*) $s(\mathbf{U}(\boldsymbol{x}'))\mathcal{R}_{\beta(\mathbf{U}(\boldsymbol{x}'))}[\mathbf{V}](\boldsymbol{x}-\boldsymbol{x}')$ is added to all tensors in a certain neighborhood, which is determined by the scale of the voting field, where $\boldsymbol{x}$ is the position where the vote is casted.

The *stick voting field* $\mathbf{V}$ is interpreted as a model for the continuation of line structures. It is a tensor field template in which the stickness (4.17) of the tensors describes the likelihood that a feature at position $\boldsymbol{x}$ belongs to the *same* line structure as a feature positioned in the center $(0,0)$ of the voting field with reference orientation $0°$. The orientation of the tensor defined in eq. (4.16) at $\boldsymbol{x}$ describes the most probable orientation of a feature at that position. In the next subsection we will describe the voting field in more detail.

Figure 4.3 shows the results on the ultrasound image of a kidney. The contour segment extraction in subimages (c) en (f) is achieved by applying thinning, and extraction of strings of connected pixels starting from the pixel with highest value that is not yet extracted, until a predefined number of pixels is extracted. The contours are more enhanced after tensor voting, which can be seen if we compare the extracted contours with and without the use of tensor voting. Clearly, tensor voting helps to extract longer and smoother contours.

Note, that the gradient magnitude will be less noisy when choosing a larger scale σ_{local}. Therefore, the *edge focussing* method [67], which detects contours by searching the most important contours on a large scale and refining the positions by tracking the edges to finer scales in scale space, might outperform the tensor voting method. However, that method is not usable for applications, in

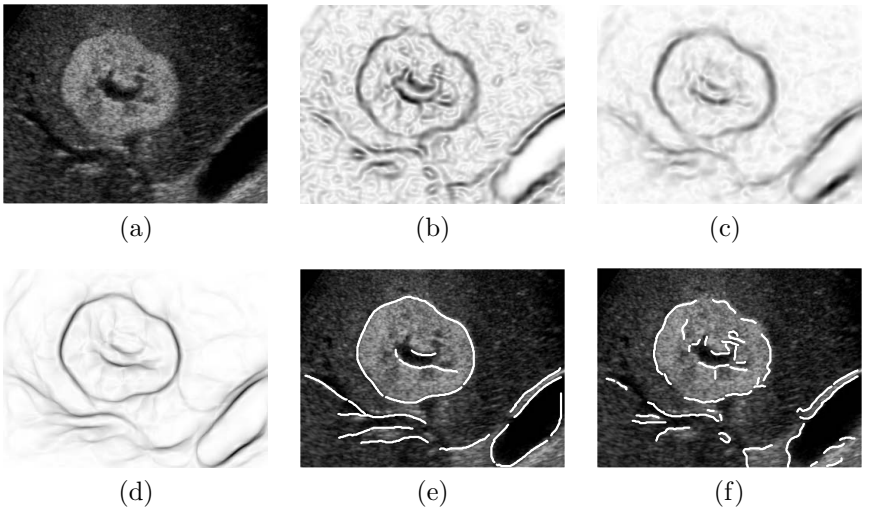


Figure 4.3: Example of tensor voting on an ultrasound image of a kidney. (a) Original image, size 345×260 pixels. (b) Gradient magnitude with $\sigma = 3$ pixels. (c) Result after first tensor voting step using the bandlimited voting field cf. Subsection 4.3.4.1 with $n = 4$ and $\sigma = 15$. (d) Result after second tensor voting step with same settings. (e) Extracted line segments using image (d). (f) Extracted line segments using image (b). In both (e) and (f), 1250 pixels were extracted for the sake of comparison.

which lines or oriented patterns have to be detected in noisy images.

4.3.2 Voting Fields

For the design of the stick voting field, Medioni et al. assume that the best connection between two points with one orientation imposed is a circular arc (Figure 4.4a). This yields a *cocircularity* pattern which is encoded in a tensor field as

$$\begin{pmatrix} \cos 2\phi \\ \sin 2\phi \end{pmatrix} \begin{pmatrix} \cos 2\phi \\ \sin 2\phi \end{pmatrix}^{\mathrm{T}} = \begin{pmatrix} (1+\cos(4\phi)) & \sin(4\phi) \\ \sin(4\phi) & (1-\cos(4\phi)) \end{pmatrix}. \tag{4.22}$$

To obtain a locally confined voting field the cocircularity pattern is modulated with a Gaussian function that decays with radius curve length and curvature. Furthermore, points above and below the main diagonals in the field are too unlikely to belong to the same structure as the point in the center of the field. These considerations yield the following voting field

$$\mathbf{V}(\boldsymbol{x}) = \Upsilon(\phi) e^{-\left(\frac{\phi r}{\sigma \sin\phi}\right)^2 - p\left(\frac{2\sigma \sin\phi}{r}\right)^2} \begin{pmatrix} (1+\cos(4\phi)) & \sin(4\phi) \\ \sin(4\phi) & (1-\cos(4\phi)) \end{pmatrix}, \tag{4.23}$$

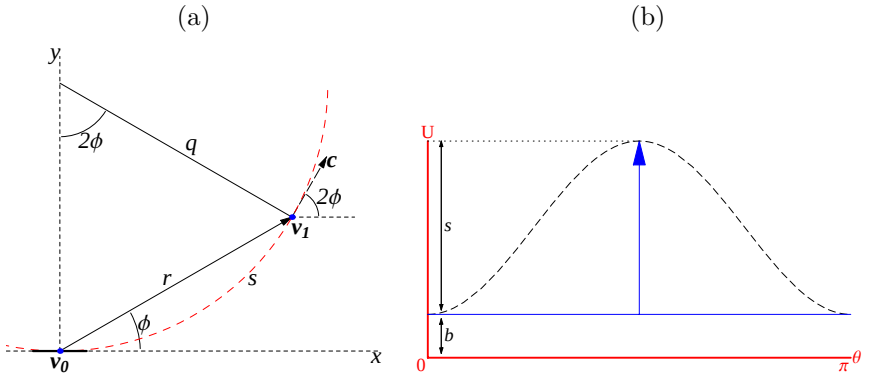


Figure 4.4: (a) Geometric reasoning for derivation of the cocircularity pattern. Position $\boldsymbol{v}_0$ has an imposed horizontal orientation, and the angle of the vector connecting $\boldsymbol{v}_0$ to $\boldsymbol{v}_1$ is ϕ, implying that $\boldsymbol{c} = (\cos 2\phi, \sin 2\phi)^{\mathrm{T}}$. (b) Illustration of the angular thinning process in tensor voting. The dashed curve shows the original angular response $A(\theta) = \boldsymbol{n}(\theta)^{\mathrm{T}} \cdot \mathbf{A} \cdot \boldsymbol{n}(\theta)$ of a 2-tensor with $\beta = \pi/2$, stickness s, and ballness b. After thinning, we obtain a constant signal b plus a delta spike at β, i.e. $b + s\,\delta(\theta - \beta)$.

where $\boldsymbol{x} = (r\cos\phi, r\sin\phi)$, $\sigma > 0$ is the scale of the voting field, p is a dimensionless constant describing the relative weight of the curvature, and

$$\Upsilon(\phi) = \begin{cases} 0 & \text{if } \frac{\pi}{4} < (\phi \mod \pi) < \frac{3\pi}{4} \\ 1 & \text{otherwise.} \end{cases} \tag{4.24}$$

The orientation-angular components of the tensors in this field are

$$\hat{V}_{m_o}(\boldsymbol{x}) = \sqrt{\frac{\pi}{2}}\,\Upsilon(\phi)\, e^{-\left(\frac{\phi r}{\sigma \sin\phi}\right)^2 - p\left(\frac{2\sigma \sin\phi}{r}\right)^2} \left(e^{-i2m_o\phi} + \delta_{m_o}\right), \tag{4.25}$$

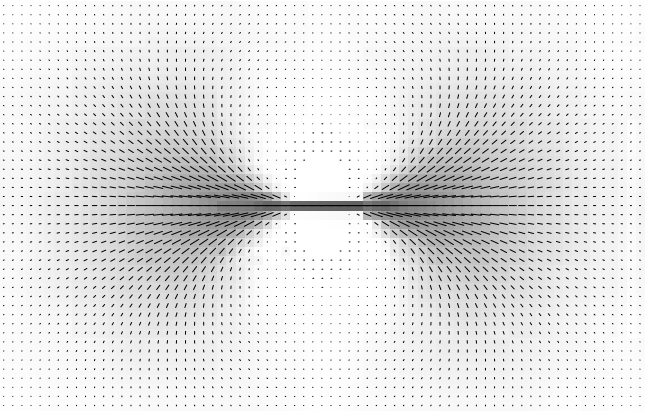
for $m_o \in \{-2, 0, 2\}$. Figure 4.5(a)a shows an example of this stick voting field.

The *ball voting field* $\mathbf{V}_{\mathrm{B}}$ is a tensor field template in which the stickness of the tensors describes the likelihood that a feature at position $\boldsymbol{x}$ belongs to the *same* line structure as the feature positioned in the center $(0, 0)$ of the voting field with *unknown* orientation. Since no initial knowledge of orientation is assumed, this voting field is required to be isotropic, i.e.

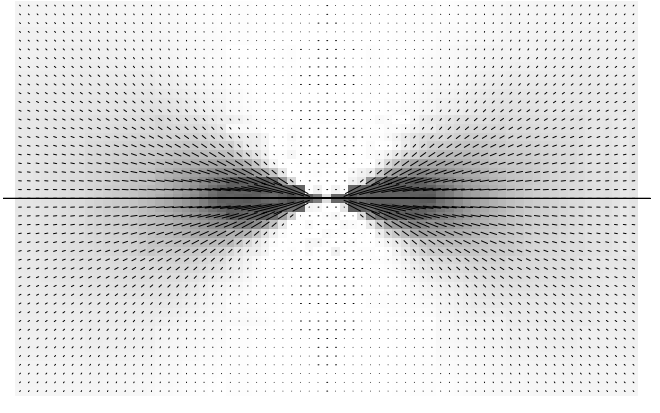
$$\mathbf{R}_\alpha \mathbf{V}_{\mathrm{B}}(\mathbf{R}_\alpha^{-1}\boldsymbol{x})\mathbf{R}_\alpha^{-1} = \mathbf{V}_{\mathrm{B}}(\boldsymbol{x}), \quad \forall\, \alpha, \boldsymbol{x}. \tag{4.26}$$

If we do not know the orientation of the local image structure, the best we can do is to average over all orientations equally. Therefore, Medioni et al. obtain a ball voting field that fulfills this requirement by integrating all rotated versions of the stick voting field $\mathbf{V}$, i.e.

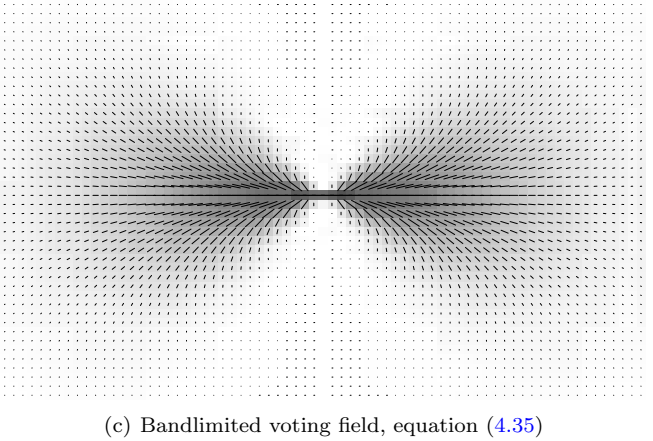
$$\mathbf{V}_{\mathrm{B}}(\boldsymbol{x}) = \frac{1}{2\pi} \int_0^{2\pi} \mathbf{R}_\alpha \mathbf{V}(\mathbf{R}_\alpha^{-1}\boldsymbol{x})\mathbf{R}_\alpha^{-1} \mathrm{d}\alpha. \tag{4.27}$$



(a) Original stick voting field, equation (4.23)



(b) Truncated stochastic completion field, even m_s-components only, equation (3.79).



(c) Bandlimited voting field, equation (4.35)

Figure 4.5: Illustration of different stick voting fields.

Medioni et al. only consider the voting field of equation (4.23). We note, however, that this is definitely not the only possible choice. One can use other voting fields that are based on various models for lines and contours. For example, in [2] the stochastic completion kernel (see Section 3.8) is embedded in the tensor voting framework for tensor voting. For this purpose the kernel is made 180°-symmetric by leaving out the odd spatial-angular frequency components and only keeping the orientation-angular frequency components $m_{\mathrm{o}} \in \{-2, 0, 2\}$. This voting field is illustrated in Figure 4.5(b). In Subsection 4.3.4.1 we will propose another possible voting field, shown in Figure 4.5(c), which has the practical benefit of being a steerable kernel.

4.3.3 Connection with Orientation Scores

It is straightforward to derive that by choosing

$$V(\boldsymbol{x}, \theta) = \frac{1}{\sqrt{2\pi}} \sum_{m_{\mathrm{o}} \in \{-2,0,2\}} (\mathbf{V}(\boldsymbol{x}), \mathbf{E}_{m_{\mathrm{o}}}) e^{i m_{\mathrm{o}} \theta}, \tag{4.28}$$

$$U(\boldsymbol{x}, \theta) = s(\mathbf{U}(\boldsymbol{x}))\delta(\theta - \beta(\mathbf{U}(\boldsymbol{x}))) + b(\mathbf{U}(\boldsymbol{x})), \tag{4.29}$$

the tensor voting operation can be written as

$$\mathbf{W}(\boldsymbol{x}) = \sum_{m_{\mathrm{o}} \in \{-2,0,2\}} \mathcal{F}_{\theta}[V *_{SE(2)} U]_{m_{\mathrm{o}}} \, \hat{\mathbf{E}}_{m_{\mathrm{o}}}. \tag{4.30}$$

In the latter equation we have expressed tensor voting in terms of an $SE(2)$-convolution, where both stick voting and ball voting are unified. Thus, tensor voting is an $SE(2)$-convolution of a lifted orientation score with an additional constant component for ball voting, cf. Subsection 3.7.3, with a 2-tensor-valued kernel, cf. Section 4.2.

We see that tensor voting encodes the orientations by means of delta spikes cf. (4.29). The orientation-angular Fourier decomposition of U gives a nonzero results for all even m_{o}, so in fact we include all harmonics of the $|m_{\mathrm{o}}| = 2$ orientation-angular frequency of the original tensor $\mathbf{U}$. This can be interpreted as an angular *thinning* operation as visualized in Figure 4.4b. Note that this thinning step is the only nonlinearity in the tensor voting framework. That is, the tensor voting operation itself cf. (4.20) is linear, but the thinning step, which is a standard (implicit) preprocessing step in the tensor voting framework, is nonlinear.

The following is also important to note. Since orientation is a periodic quantity with periodicity π, the natural way to calculate an average orientation $\overline{\beta}$ of an ensemble of orientations β_i, $i = 1 \ldots N$, with weightings w_i is

$$\overline{\beta} = \frac{1}{2} \arg \sum_{i=1}^{N} w_i e^{2i\,\beta_i}. \tag{4.31}$$

We can observe that this equation is similar to the tensor voting operation for $|m_{\mathrm{o}}| = 2$, so the context-enhanced orientation $\beta(\mathbf{U})$ in tensor voting is in fact the weighted *average* orientation of all incoming votes.

4.3.4 Steerable Tensor Voting

A technical difficulty in implementing tensor voting is the orientational alignment of the voting field with the stickness tensor, which needs to be done at every position. However, we can derive a *steerable tensor voting* method by applying the results of Sections 3.5 and 3.7. In earlier work [51], we have provided a more direct derivation of steerable tensor voting, without the intermediate step of writing tensor voting as an $SE(2)$-convolution. However, it is more insightful to link tensor voting to orientation scores and $SE(2)$-convolutions.

The voting field $\mathbf{V}$ is steerable if we can write it as

$$\mathbf{V}(\boldsymbol{x}) = \sum_{m_{\mathrm{o}} \in \{-2,0,2\}} \sum_{m_{\mathrm{s}}=-M_{\mathrm{s}}}^{M_{\mathrm{s}}} \mathcal{C}[V]_{m_{\mathrm{o}},m_{\mathrm{s}}}(\boldsymbol{x}) \mathbf{E}_{m_{\mathrm{o}}}, \tag{4.32}$$

where the relation between $\mathbf{V}$ and V is given in (4.28). The operator to obtain steerable components $\mathcal{C}$ is defined in (3.44) on page 63. The steerable components $\mathcal{C}[V]_{m_{\mathrm{o}},m_{\mathrm{s}}}$ can be used directly in equation (3.49)

$$\mathbf{W}(\boldsymbol{x}) = \sqrt{2\pi} \sum_{m_{\mathrm{o}} \in \{-2,0,2\}} \left(\sum_{m_{\mathrm{s}}=-M_{\mathrm{s}}}^{M_{\mathrm{s}}} (\mathcal{C}[V]_{m_{\mathrm{o}},m_{\mathrm{s}}} *_{\mathbb{R}^2} \mathcal{F}_\theta[U]_{m_{\mathrm{s}}})(\boldsymbol{x}) \right) \hat{\mathbf{E}}_{m_{\mathrm{o}}}, \tag{4.33}$$

where $\mathcal{F}_\theta[U]_{m_{\mathrm{s}}}$ is the orientation-angular Fourier decomposition of U, given by

$$\mathcal{F}_\theta[U]_{m_{\mathrm{s}}}(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi}} s(\mathbf{U}(\boldsymbol{x})) e^{-i(m_{\mathrm{s}})\beta(\mathbf{U}(\boldsymbol{x}))} + \delta_{m_{\mathrm{s}}} b(\mathbf{U}(\boldsymbol{x})). \tag{4.34}$$

So we can apply tensor voting simply by calculating a number of two-dimensional complex-valued convolutions.

A stick voting field should be 180°-symmetric, because the orientation $\tilde{\beta}$ encodes orientations modulo π, meaning that all odd spatial-angular frequency components are zero. Furthermore, taking into account that $\mathcal{F}_\theta[U]_2 = \overline{\mathcal{F}_\theta[U]_{-2}}$ we need at most $2(M_{\mathrm{s}} + 1)$ 2D convolutions.

The direct implementation of steerable tensor voting consists of the following steps

1. For each value of m_{s} with non-zero steerable components $\mathcal{C}[V]_{m_{\mathrm{o}},m_{\mathrm{s}}}$, calculate (4.34): $\mathcal{O}\left(C_s N_s^2\right)$;

2. Apply an $\mathbb{R}^2$-convolution for each value of m_s and $m_\mathrm{o} \in \{-2, 0, 2\}$ and sum the results, cf. (4.33): $\mathcal{O}\left(C_s N_s^2 K_s^2\right)$.

The FFT implementation of steerable tensor voting consists of the following steps

1. For each value of m_s with non-zero steerable component(s) $\mathcal{C}[V]_{m_\mathrm{o},m_\mathrm{s}}$, calculate (4.34) and apply an FFT: $\mathcal{O}\left(C_s N_s^2 \log N_s\right)$;
2. Multiplication in the Fourier domain for each value of m_s and $m_\mathrm{o} \in \{-2, 0, 2\}$, and sum the results: $\mathcal{O}\left(C_s N_s^2\right)$;
3. Inverse 2D FFT for each $m_\mathrm{o} \in \{-2, 0, 2\}$: $\mathcal{O}\left(N_s^2 \log N_s\right)$.

Summarizing, the complexities for steerable tensor voting are $\mathcal{O}\left(C_s N_s^2 K_s^2\right)$ for a direct implementation and $\mathcal{O}\left(C_s N_s^2 \log N_s\right)$ for an FFT implementation.

4.3.4.1 Example Steerable Voting Field

The voting field as proposed by Medioni et al. is not a steerable filter, because it contains infinitely high spatial-angular frequency components. Therefore it has to be truncated to use it with steerable tensor voting. As an example, below we will illustrate how to derive a simpler steerable voting field, which shows good performance in most practical situations.

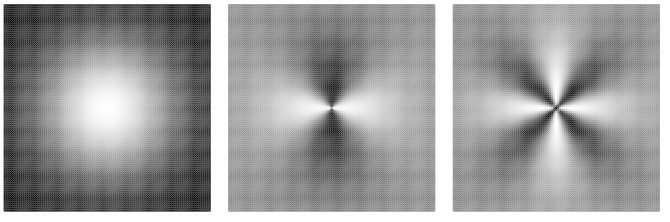
Similar to Medioni's voting field, we assume that the best connection between two points with one orientation imposed is a circular arc (Figure 4.4). Now we modulate the cocircularity pattern with a Gaussian function that decays with radius r. To penalize high-curvature arcs, we modulate with the term $\cos^{2n}\phi$, where $n \in \mathbb{N}$ is a parameter specifying the speed of decay of the field as function of ϕ. This angular decay function is similar to the one for instance used in [69] and has the useful property that it is frequency spectrum is bandlimited by $2n$. The voting field now becomes

$$\mathbf{V}(\boldsymbol{x}) = \frac{1}{N} e^{\frac{-r^2}{2\sigma^2}} \cos^{2n}(\phi) \begin{pmatrix} (1+\cos(4\phi)) & \sin(4\phi) \\ \sin(4\phi) & (1-\cos(4\phi)) \end{pmatrix}, \tag{4.35}$$

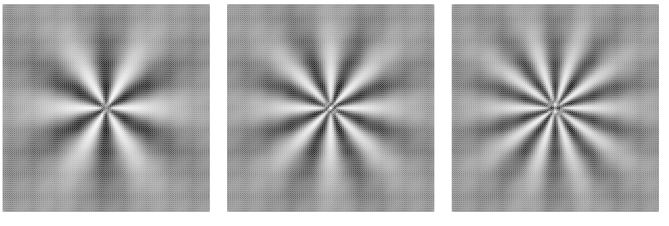
where $\boldsymbol{x} = (r\cos\phi, r\sin\phi)$, $\sigma \in \mathbb{R}^+$ is the scale of the voting field, and N is a normalization factor. In the following, to get simpler equations, we will use $N = \sqrt{2\pi}/16$ and $n = 2$.

We apply full-angular Fourier decomposition on this voting field, which after some trigonometric calculations yields

$$\begin{pmatrix} \hat{\mathbf{V}}_{-2}(\boldsymbol{x}) \\ \hat{\mathbf{V}}_{0}(\boldsymbol{x}) \\ \hat{\mathbf{V}}_{2}(\boldsymbol{x}) \end{pmatrix} = e^{-\frac{r^2}{2\sigma^2}} \begin{pmatrix} 1 + 4e^{i2\phi} + 6e^{i4\phi} + 4e^{i6\phi} + e^{i8\phi} \\ e^{-i4\phi} + 4e^{-i2\phi} + 6 + 4e^{i2\phi} + e^{i4\phi} \\ e^{-i8\phi} + 4e^{-i6\phi} + 6e^{-i4\phi} + 4e^{-i2\phi} + 1 \end{pmatrix}. \tag{4.36}$$



(a) $m_{\mathrm{s}} = 0$ (b) $m_{\mathrm{s}} = 2$ (c) $m_{\mathrm{s}} = 4$



(d) $m_{\mathrm{s}} = 6$ (e) $m_{\mathrm{s}} = 8$ (f) $m_{\mathrm{s}} = 10$

Figure 4.6: Illustration of the real part of the basis functions $w_{m_{\mathrm{s}}}$ (see (4.38)) for $m_{\mathrm{s}} = 0, 2, 4, 6, 8, 10$.

In this case, for every orientation-angular frequency component m_{o} we have 5 spatial-angular frequency component m_{s}, and all radial functions are the same. We can now write this filter in a simplified steerable form

$$\begin{pmatrix} \mathcal{R}_\alpha[\hat{\mathbf{V}}_{-2}](\boldsymbol{x}) \\ \mathcal{R}_\alpha[\hat{\mathbf{V}}_{0}](\boldsymbol{x}) \\ \mathcal{R}_\alpha[\hat{\mathbf{V}}_{2}](\boldsymbol{x}) \end{pmatrix} = \begin{pmatrix} 0 & 0 & e^{6i\alpha} \\ 0 & 0 & 4e^{4i\alpha} \\ 0 & e^{4i\alpha} & 6e^{2i\alpha} \\ 0 & 4e^{2i\alpha} & 4 \\ e^{2i\alpha} & 6 & e^{-2i\alpha} \\ 4 & 4e^{-2i\alpha} & 0 \\ 6e^{-2i\alpha} & e^{-4i\alpha} & 0 \\ 4e^{-4i\alpha} & 0 & 0 \\ e^{-6i\alpha} & 0 & 0 \end{pmatrix}^{\mathrm{T}} \begin{pmatrix} v_{-8}(\boldsymbol{x}) \\ v_{-6}(\boldsymbol{x}) \\ \vdots \\ v_{6}(\boldsymbol{x}) \\ v_{8}(\boldsymbol{x}) \end{pmatrix}, \tag{4.37}$$

where the matrix contains the linear coefficients as function of rotation, and the vector at the right side contains the basis filters. The basis filters are shown in Figure 4.6 and they are given by

$$v_{m_{\mathrm{s}}}(\boldsymbol{x}) = e^{-\frac{r^2}{2\sigma^2}} e^{im_{\mathrm{s}}\phi} = e^{-\frac{x^2+y^2}{2\sigma^2}} \left(\frac{x+iy}{\sqrt{x^2+y^2}} \right)^{m_{\mathrm{s}}}, \quad \text{for } \boldsymbol{x} \neq (0,0). \tag{4.38}$$

These filter kernels $w_{m_{\mathrm{s}}}(\boldsymbol{x})$ must be tabelized for $m_{\mathrm{s}} = 0, 2, 4, 6, 8$. Given the stickness s and orientation β of tensors in $\mathbf{U}$, we need to calculate a number of complex-valued feature images $\hat{U}_m = \mathcal{F}_\theta[U]_m$ where U is defined in (4.29), for

$m = 0, 2, 4, 6$ where $m = m_{\mathrm{o}} + m_{\mathrm{s}}$. This yields

$$\begin{aligned}\hat{W}_{-2}(\boldsymbol{x}) = \overline{\hat{W}_2}(\boldsymbol{x}) = \sqrt{2\pi}\big((v_0 * \overline{\hat{U}_2}) + 4(v_2 * \hat{U}_0) + 6(v_4 * \hat{U}_2) \\ + 4(v_6 * \hat{U}_4) + (v_8 * \hat{U}_6)\big), \\ \hat{W}_0(\boldsymbol{x}) = \sqrt{2\pi}\left(\mathrm{Re}\big(6(v_0 * \hat{U}_0) + 8(v_2 * \hat{U}_2) + 2(v_4 * \hat{U}_4)\big)\right).\end{aligned} \tag{4.39}$$

We see that steerable tensor voting with this voting field in total requires 8 two-dimensional convolutions.

4.3.5 First Order Voting and Higher Orientation-Angular Frequencies

Normal tensor voting only votes with 2-tensors, so $m_{\mathrm{o}} \in \{-2, 0, 2\}$. Tong et al. [133] propose an extension where besides voting with 2-tensors, there is a separate process that votes with 1-tensors, i.e. vectors that are called *polarity vectors*. The idea is that the resulting vectors give information on the direction from which the majority of the 2-tensor votes come. This can be used for instance as a line end-point detector, since at a line end-point the majority of the votes will come from the direction in which the line propagates.

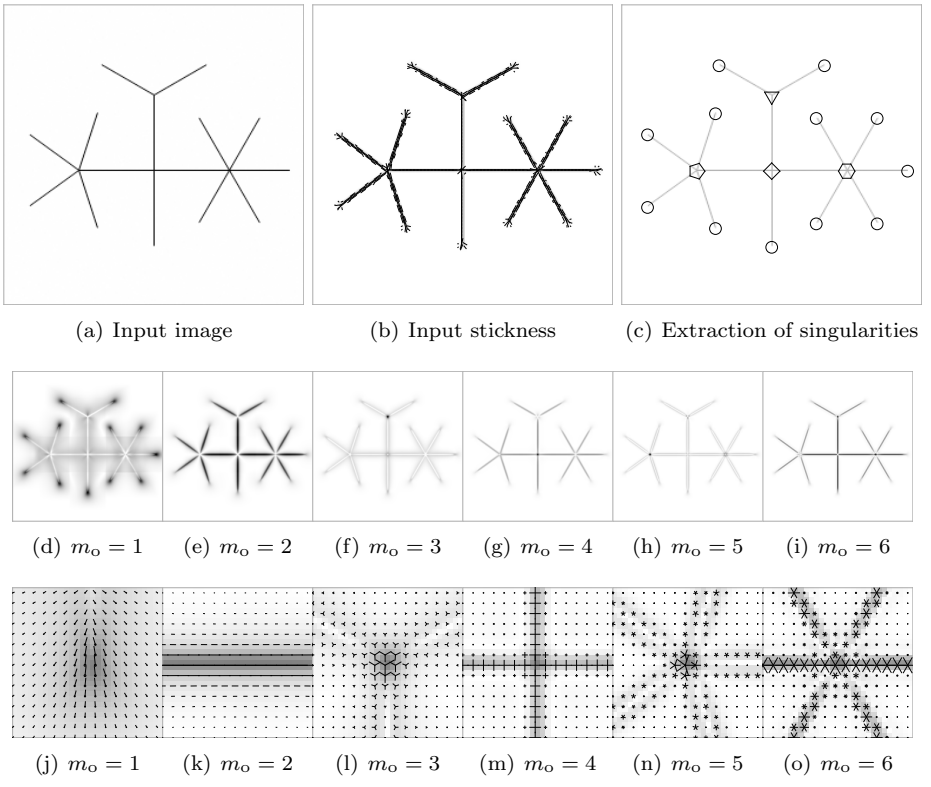
The operational definition is analogous to the left integral of equation (4.20), but $\mathbf{V}$ and $\mathbf{W}$ are vectors instead. The same voting field can be used, where the cocircularity is encoded in vectors $(\cos 2\phi, \sin 2\phi)^{\mathrm{T}}$.

In our framework, first order voting can be embedded straightforwardly. A vector $\boldsymbol{B} = (b_1, b_2)^{\mathrm{T}}$ has angular frequency components $m_{\mathrm{o}} \in \{-1, 1\}$ with the following relationships

$$\begin{aligned}\boldsymbol{B} = B_{-1}\boldsymbol{e}_{-1} + B_1\boldsymbol{e}_1, \quad B_{-1} = b_1 + ib_2, B_1 = b_1 - ib_2, \\ \boldsymbol{e}_{-1} = \frac{1}{2}\begin{pmatrix}1\\-i\end{pmatrix}, \quad \boldsymbol{e}_1 = \frac{1}{2}\begin{pmatrix}1\\i\end{pmatrix}.\end{aligned} \tag{4.40}$$

The vector voting field is given by equation (4.25), but now with $m_{\mathrm{o}} \in \{-1, 1\}$. First order tensor voting is obtained by equation (4.30) with $m_{\mathrm{o}} \in \{-1, 1\}$. So in conclusion, tensor voting with both 1-tensors and 2-tensors can be viewed as an $SE(2)$-convolution with as result an orientation score with orientation-angular components $m_{\mathrm{o}} \in \{-2, -1, 0, 1, 2\}$.

We can straightforwardly include higher m_{o}-components as well. In our framework, the meaning of these additional components is immediately clear: it results in an orientation score with a higher angular resolution. In terms of tensors, an orientation score that is angularly bandlimited by M_{o} can be mapped 1-to-1 to a symmetric M_{o}-tensor. Such tensor representation, however, is much less



(a) Input image (b) Input stickness (c) Extraction of singularities

(d) $m_o = 1$ (e) $m_o = 2$ (f) $m_o = 3$ (g) $m_o = 4$ (h) $m_o = 5$ (i) $m_o = 6$

(j) $m_o = 1$ (k) $m_o = 2$ (l) $m_o = 3$ (m) $m_o = 4$ (n) $m_o = 5$ (o) $m_o = 6$

Figure 4.7: Example of tensor voting with the stochastic completion field (see Section 3.8), which contains higher m_o-components, making it possible to infer higher angular frequency information. (a) Input artificial image. Size 256×256 pixels. (b) Input stickness s obtained using the Hessian of Gaussian derivatives, with superimposed orientation field. This stickness and orientation map are the input for an $SE(2)$-convolution (or in other words, a tensor voting step with higher orientation-angular frequencies) with the stochastic completion field cf. (3.78) on page 76 with $\sigma = 0.2$ and $\lambda = 0$. (c) Resulting extraction of singularities; the circles indicate the 12 strongest local maxima found in the image of (d), which turns out to be a good end-point detection method. The triangle, diamond, pentagon, and hexagon show the strongest local maxima found in the images (f), (g), (h), and (i) respectively, and turn out to be crossing detectors where m lines cross. (c)-(i) responses $|\hat{U}_{m_o}|$. $m_o = 1$ in (d) is similar to the length of the polarity vectors as described in [133] and is high on end-points. $m_o = 2$ in (e) is the stickness $s = |\hat{U}_{m_o=2}(\boldsymbol{x})|$ and is high on line structures. The components $m_o \geq 3$ are useful to extract information on different types of crossings of line structures, as is done for figure (c). (j)-(o) Zoomed in singularity points, where the superimposed "m_o-glyphs" show the estimated phase (encoded by the orientation of the line segments) and the absolute value (encoded by the length).

intuitive to work with and therefore we favor to consider the cases $M_{\mathrm{o}} > 2$ as orientation scores. Figure 4.7 shows an example of using the higher values of m_{o}. Although the input consists of stickness, ballness and orientation only, the result with $m_{\mathrm{o}} \in \{1, 2, 3, 4, 5, 6\}$ supplies us with much richer information for instance for the detection of e.g. crossings.

4.4 Relation to the Structure Tensor

Another widespread image processing tool using tensors is the structure tensor. In this section we will show how this method can be expressed as an $SE(2)$-convolution.

The "standard" structure tensor [50] is obtained by first calculating the first order Gaussian derivatives, $\nabla f = ((\partial_x G_\sigma) * f\,,\, (\partial_y G_\sigma) * f)^{\mathrm{T}}$. By taking the dyadic product $\mathbf{U}(\boldsymbol{x}) = \nabla f(\boldsymbol{x})\, \nabla f(\boldsymbol{x})^{\mathrm{T}}$ one obtains a 2×2 matrix for each pixel with one positive eigenvalue $\lambda_1(\boldsymbol{x}) = \|\nabla f(\boldsymbol{x})\|$, with corresponding eigenvector $\nabla f(\boldsymbol{x}) / \|\nabla f(\boldsymbol{x})\|$ and one zero eigenvalue. The tensor field $\mathbf{U}$ is subsequently convolved componentwise with an isotropic Gaussian kernel with scale ρ, $\mathbf{W} = \mathbf{U} * G_\rho$. The result is a tensor field containing richer information on edges and junctions.

The structure tensor operation is rewritten as an $SE(2)$-convolution as follows

$$\Psi(\boldsymbol{x}, \theta) = G_\rho(\boldsymbol{x}) \cos^2 \theta, \qquad U(\boldsymbol{x}, \theta) = ||\nabla f(\boldsymbol{x})||^2 \delta(\theta - \angle(\nabla f(\boldsymbol{x}))), \quad (4.41)$$

$$\mathbf{W} = \textstyle\sum_{m_{\mathrm{o}} \in \{-2, 0, 2\}} \mathcal{F}_\theta [\Psi *_{SE(2)} U]_{m_{\mathrm{o}}}(\boldsymbol{x})\, \hat{\mathbf{E}}_{m_{\mathrm{o}}}, \quad (4.42)$$

where "$\angle$" denotes the angle of the gradient ∇f relative to the horizontal direction.

Inspired by tensor voting, Köthe [86] proposes to use an anisotropic smoothing filter for the structure tensor to prevent spurious blurring of the structure tensor. He proposes to use a so-called "hour-glass" filter, which is given by

$$\Psi_{\mathrm{S}} = \frac{1}{N} e^{\frac{-r^2}{2\sigma'^2}} e^{\frac{-\tan(\phi)}{2\rho^2}}, \quad \boldsymbol{x} = (r \cos\phi, r \sin\phi). \quad (4.43)$$

The tensor field $\mathbf{U}$ is smoothed componentwise with this kernel, which is at every position aligned with the edge orientation $\beta = \angle(\nabla f(\boldsymbol{x}))$. The equation becomes

$$\mathbf{W}(\boldsymbol{x}) = \int_{\mathbb{R}^2} \Psi_{\mathrm{S}}(\mathbf{R}^{-1}_{\beta(\boldsymbol{x}')}(\boldsymbol{x} - \boldsymbol{x}'))\mathbf{U}(\boldsymbol{x}')\mathrm{d}\boldsymbol{x}'. \quad (4.44)$$

Again it is possible to express this operation in terms of an $SE(2)$-convolution, by choosing

$$\Psi(\boldsymbol{x}, \theta) = \Psi_{\mathrm{S}}(\boldsymbol{x}) \cos^2 \theta, \qquad U(\boldsymbol{x}, \theta) = ||\nabla f(\boldsymbol{x})||^2 \delta(\theta - \beta(\boldsymbol{x})), \quad (4.45)$$

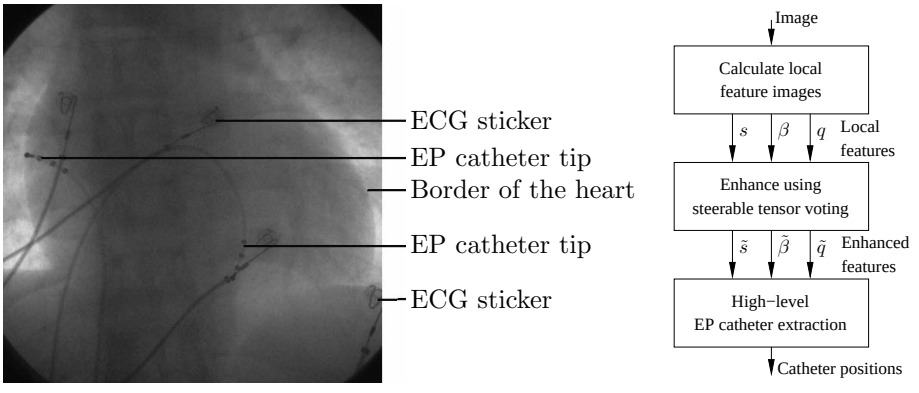
and using (4.42). This resembles the special case of separable kernels with a lifted orientation score as input, see Subsection 3.7.3.

Nonlinear structure tensors [15] are obtained by applying a nonlinear diffusion on the tensor components instead of a linear diffusion. Since this approach is nonlinear it cannot be expressed as an $SE(2)$-convolution. Furthermore, the nonlinear diffusivity function that is used for adaptive componentwise diffusion of the tensor image is not rotation invariant, since it is constructed by a nonlinear combination of componentwise spatial derivatives on the tensor image. This implies that the method does not adhere to the $SE(2)$ group structure. One can, however, straightforwardly develop a nonlinear structure tensor approach which does adhere to the $SE(2)$ group. The tools for this purpose are provided by Subsection 2.9.1, this chapter, and Chapter 6.

4.5 Medical Application: EP Catheter Extraction with Tensor Voting

In this section we will apply parts of the results of this chapter to a medical application. Cardiac catheter ablation is a procedure to treat heart rhythm disorders (arrhythmias). It involves the insertion of one or more flexible thin tubes, called *electrophysiology (EP) catheters*, through small skin incisions, usually in the groin. These catheters are threaded through blood vessels into the heart. The EP catheters contain a number of electrodes used to make *intracardiac electrograms*. Using these electrograms, the firing spot or conduction path causing the arrhythmias can be identified. A special EP catheter (the *ablation catheter*) emits radiofrequency energy to destroy the spot or to block the undesired conduction path, so that only the normal electrical conduction path remains. The movement of the catheter through the body is guided using a real-time X-ray fluoroscopy imaging system (Figure 4.8a).

Catheter ablation is a time-consuming medical procedure, therefore tools to speed up the process are of great interest. For this purpose, an important tool is the automation of cardiac mapping, i.e. creating a 3D map of cardiac activation patterns over the entire heart. By using bi-plane fluoroscopy, the 3D position of the catheters can be estimated and can be used to superimpose the cardiac activation sequences onto fluoroscopic images. Different research groups and companies are working on this problem, see e.g. [89, 44, 16]. In these papers, segmentation of the catheters, and especially the electrodes, is considered an important but difficult task to automate. Kynot et al. [89] have proposed an algorithm to detect the electrodes of the catheters, but problems remain in associating the electrodes with the catheters in the image. De Buck et al. [16] constructed an advanced cardiac mapping system, but this still requires the user to perform a manual segmentation of the catheter and the electrodes. Fallavollita et al. [44] developed a



(a) (b)

Figure 4.8: (a) An example of a typical EP X-ray fluoroscopy image acquired during a clinical intervention. We only want to detect the EP catheters, not the other visible elongated structures. (b) Framework of our method. See text for details.

catheter tip detection algorithm based on thresholding of the X-ray image, but due to the noisy nature of fluoroscopy images, the performance is not satisfactory.

In this section we propose a method for automatic detection of EP catheters in noisy X-ray fluoroscopy images, without the need for any user intervention. Our method detects both the catheter bodies and the corresponding electrodes. We restrict ourselves to the detection of catheters in a still image, i.e. only spatial information is used. A more detailed report can be found in [52].

Figure 4.8b shows the general framework of our EP catheter extraction process. The method is divided into three main stages. In the first stage, *calculate local feature images*, we perform preprocessing and use local filtering operations to calculate a number of *local feature images*. Because fluoroscopy images are noisy, these local feature images are unreliable. Therefore, the idea behind the next step is to use information from a larger spatial neighborhood, compared to the neighborhood of the local filters, to make the feature images more consistent and specifically enhance the elongated structures. For that purpose we use steerable tensor voting as described in Subsection 4.3.4. In the last stage, *high-level EP catheter extraction*, the enhanced feature images generated by the previous step are used to finally decide where the EP catheters are located. EP catheter-specific properties are used to discriminate the catheters from other line structures, for example using the fact that the tip contains a number of electrodes.

These three stages will be explained in the next subsections. Finally, we will evaluate the results on clinical images.

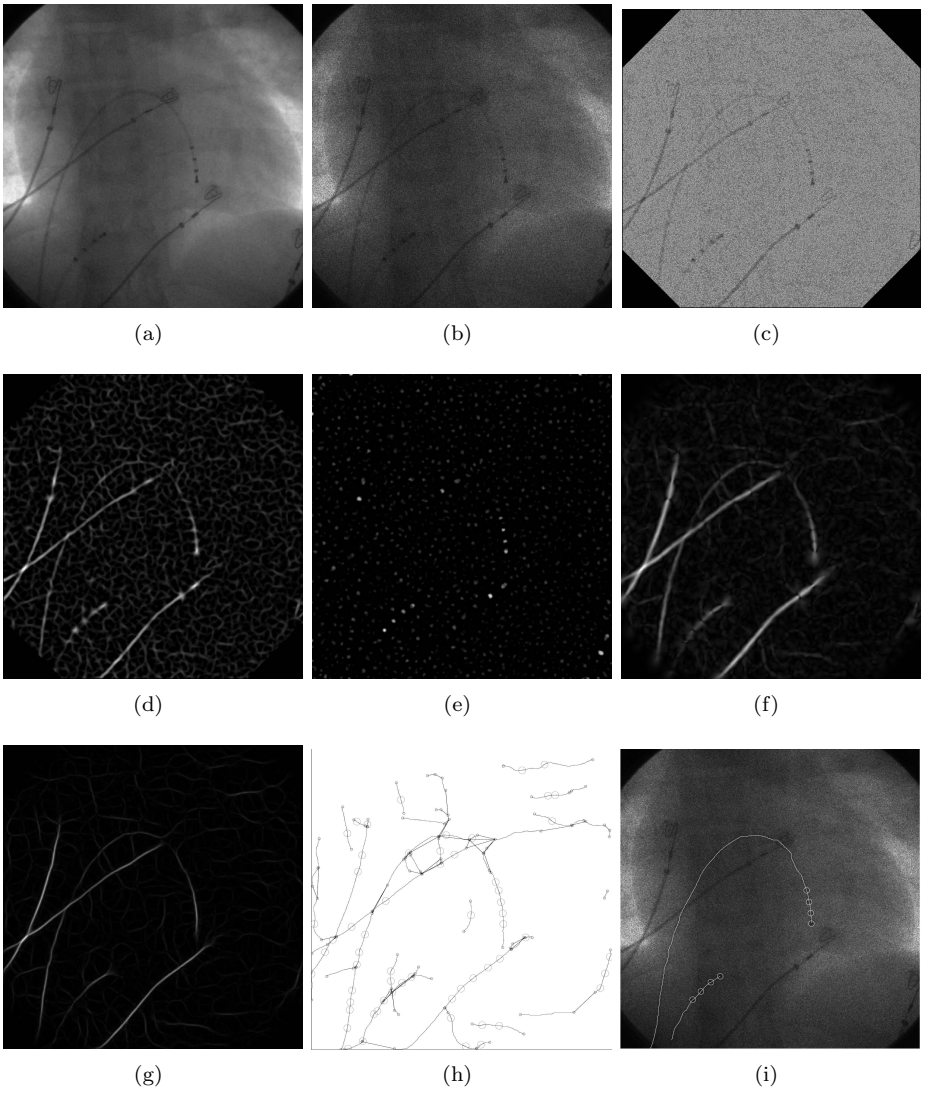


Figure 4.9: EP catheter extraction example. (a) Original image. (b) Original image with additional noise, used as input for this example. (c) Background equalized image. (d) Local ridgeness image. (e) Blobness image. (f) Result of first *tensor voting* step. (g) Result of a second tensor voting step. (h) Extracted paths (visualized by lines with small circles to indicate the end-points) and electrode candidates (visualized by the larger circles). (i) Final extraction result.

4.5.1 Local Feature Detection

Prior to any filtering we first apply background equalization to remove disturbing structures in the background. We apply a morphological closing operation with a disc-shaped structure element on a slightly blurred version of the input image, to get an image that only contains background structures. Each pixel of the original image is divided by the corresponding pixel in this image to cancel out the background structures.

The second order differential structure of an image gives important information about line-like and blob-like structures. Therefore, we construct a Hessian matrix for each pixel position by calculating second order Gaussian derivatives, see e.g. [66]. We use the 2 eigenvalues λ_1 and λ_2 with $\lambda_1 > \lambda_2$ and corresponding eigenvectors $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ to obtain the following feature images

- A local *ridgeness* image $s(x, y) = \max(\lambda_1(x, y), 0)$ (see Figure 4.9d), indicating the likelihood that the image contains a line segment at position (x, y). We use λ_1 because its value exactly corresponds to the value one would find when seeking the maximum response of the second order derivative applied in all different orientations. We only keep positive values because we know that catheters are always dark relative to the background.
- A local *orientation* image $\beta(x, y) = \angle \boldsymbol{e}_1(x, y)$ indicating the most likely orientation of a line segment at position (x, y). The orientation of the first eigenvector corresponds to the orientation at which the response of the second order derivative is maximum.
- A local *blobness* image $q(x, y) = \max(\lambda_2(x, y), 0)$ (see Figure 4.9e), indicating the likelihood that the image contains a blob-like structure at position (x, y). The motivation is that $\lambda_2 > 0$ implies that $\lambda_1 > \lambda_2 > 0$, which corresponds to a locally concave shape.

4.5.2 Contextual Enhancement by Steerable Tensor Voting

To enhance the noisy local ridgeness and orientation measures, we use steerable tensor voting described in Subsection 4.3.4. We use the steerable voting field described in Subsection 4.3.4.1.

In the EP catheter detection algorithm, we perform two subsequent tensor voting steps. The first one is performed on the local feature images s and β. From the eigenvalues $\tilde{\lambda}_1, \tilde{\lambda}_2$ and eigenvectors $\tilde{\boldsymbol{e}}_1, \tilde{\boldsymbol{e}}_2$ of the tensors in the resulting tensor image, we calculate enhanced feature images $\tilde{s}$, $\tilde{\beta}$, and $\tilde{q}$, as follows (omitting spatial coordinates for simplicity)

$$\tilde{s} = \tilde{\lambda}_1 - \tilde{\lambda}_2, \qquad \tilde{\beta} = \angle \tilde{\boldsymbol{e}}_1, \qquad \tilde{q} = q \cdot (\tilde{\lambda}_1 - \tilde{\lambda}_2). \tag{4.46}$$

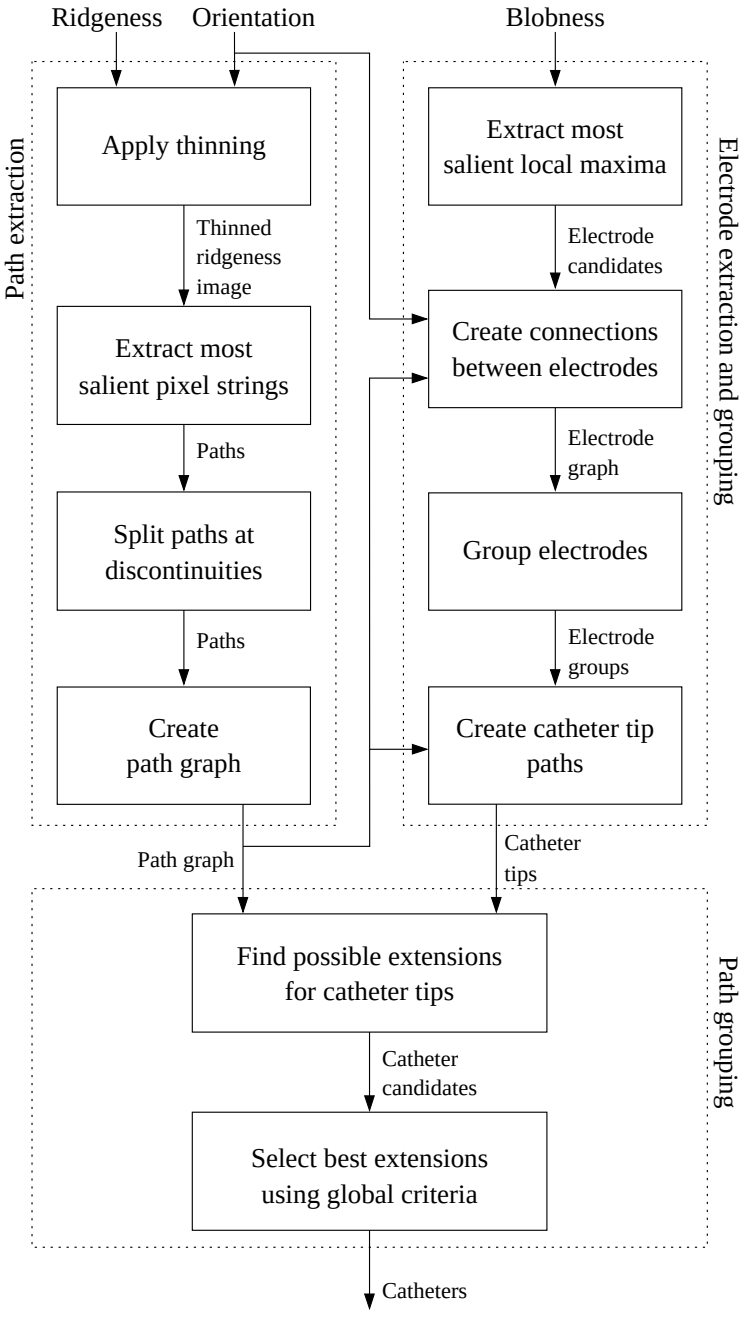


Figure 4.10: Schematic overview of the high-level extraction method.

The resulting enhancement is not always sufficient. To get more consistent curves, directional non-maximum suppression (thinning) is applied on the resulting stickness image to keep the centerlines of the curves, followed by a second tensor voting step with the thinned image as input, see Figure 4.9f and g. Thinning means that only tensors are kept that are maximum relative to their neighbors in the directions perpendicular to the local orientation $\tilde{\beta}$, i.e. all other tensors are made zero.

4.5.3 High-Level Extraction of the EP Catheters

In the last part of the algorithm, we use specific knowledge about EP-catheters to extract them. We will explain it briefly here, and refer to Figure 4.10 for a schematic overview.

The algorithm consists of three modules. The first module is *path extraction*. The ridgeness and orientation images $\tilde{s}$ and $\tilde{\beta}$ are used to perform directional non-maximum suppression, resulting in an image with curves of 1 pixel thickness. From this image we extract a number of most salient connected pixel strings (the paths). If a path exhibits very high curvature or is likely to be part of branching lines, the path is split to allow proper reconnection in the subsequent steps. From the resulting paths, a *path graph* is created (see Figure 4.9h), which has connections between paths whose spatial positions and orientations make it probable that they belong to the same global elongated object in the image.

The second module is *electrode extraction and grouping*. From the blobness image $\tilde{q}$ the most salient local maxima are extracted as electrode candidates (see Figure 4.9h). Using the extracted paths and knowledge of typical distances between electrodes, a graph is created with connections between all candidates that might be neighboring electrodes on a catheter. Then, we scan for groups of connected electrodes in this graph that have the best match with the known properties of electrode spacing on EP catheters. These electrode groups and the extracted paths are used to create *catheter tip paths*, describing the curves in the image that connect all electrodes of a catheter with each other.

The catheter tip is now fully extracted, which might already be enough for most applications. However, in the third module, *path grouping*, our algorithm also attempts to extract the entire catheter. Using the catheter tips and the path graph, different reasonable extensions of the catheter are considered. The best extension is selected based on a global criterion involving minimization of curvature and change of curvature.

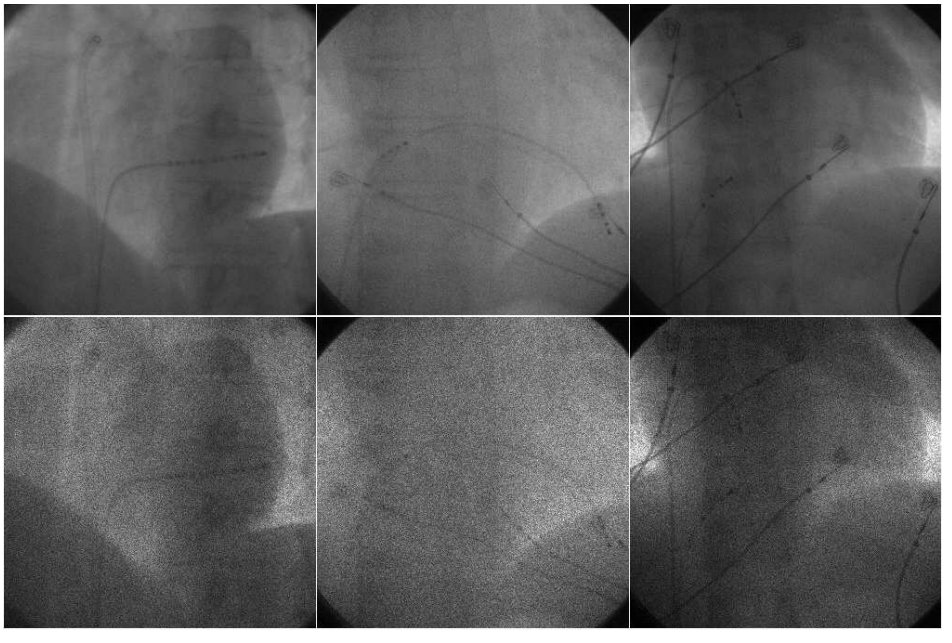


Figure 4.11: Three example images from the clinical the test set. Upper row: without additional noise. Lower row: with artificially added multiplicative Poisson noise, which is added to investigate the noise robustness.

4.5.4 Results

We implemented the EP catheter extraction algorithm partly in Mathematica and partly in C++. The parameters for local feature detection and tensor voting were optimized on 6 different images using the *signal-to-background ratio* as criterion. The parameter values are (512×512 pixel images): scale of Gaussian derivatives $\sigma_{\text{local}} = 3.4$ pixels, angular specificity of the voting field $\nu = 4$, scale of the voting field $\sigma = 15$ pixels, and scale of the voting field for the second tensor voting step $\sigma_{\text{ctx2}} = 7.5$ pixels. The parameters of the high-level extraction part were optimized using a test set of 10 images. Since these test sets are small, we think that the parameters can be further optimized.

We used an evaluation set of 50 randomly selected X-ray images acquired during 4 clinical interventions (see Figure 4.11), without any image that was used for parameter tuning. These images contain 103 EP catheters that all contain from 4 up to 10 electrodes. The catheters were extracted both with tensor voting and without tensor voting (by simply skipping this step) and both with and without added multiplicative Poisson noise.

Each catheter extraction result was assigned to one of the following categories:

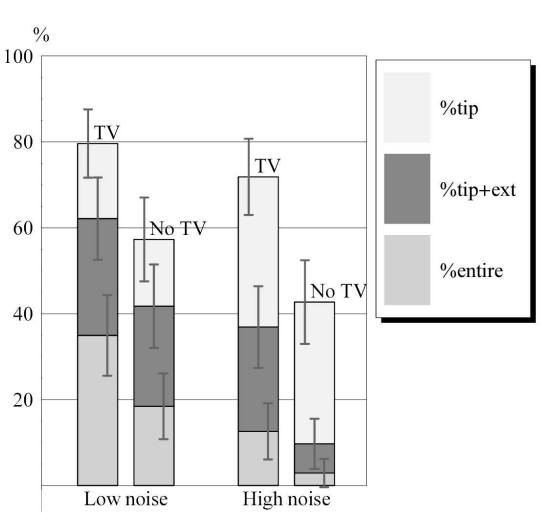


Figure 4.12: Extraction results on the test set for low noise and high noise images, and with and without tensor voting. The colours indicate extracted catheter tips (%tip), extracted additional catheter segments (%tip+ext), and extracted entire catheters (% entire). The grey vertical lines with horizontal serifs indicate confidence intervals of 95%.

(1) catheter not detected at all, (2) successful extraction of the tip only, i.e. the part containing the electrodes, (3) successful extraction of the tip plus an additional catheter segment with the same length as the tip part, and (4) successful extraction of the entire catheter.

Figure 4.12 displays the results of the catheter extraction, showing that especially the tip detection is successful. The increase in performance due to tensor voting is high. For example, the success rate of catheter tip extraction increased from 57% to 80% and from 43% to 72% for low and high noise images respectively. The success rates on extraction of tip+extension and extraction of the entire catheter are still low, especially in noisy images. For clinical practice, however, tip extraction is most relevant. Figure 4.9 shows an example of the entire EP catheter extraction algorithm.

4.5.5 Discussion

In this section we introduced an algorithm for the extraction of EP catheters in fluoroscopy images. One novelty is the use of steerable tensor voting to enhance the noisy curves in the images, which results in more noise robustness, as shown in the evaluation. A further novelty is that we are able to extract the EP catheter

fully automatically, without an initial seed, by using an advanced EP catheter-specific high-level extraction algorithm.

The implementation is currently still too slow for clinical use. This can be improved by e.g. making faster implementation, for instance on the GPU or an FPGA. Finally, it should also be noted that both the extraction results and the computational performance could be greatly improved by including the information from previous frames in the image sequence.

4.6 Conclusions

We showed that orientation scores provide an intuitive framework for tensor-valued image processing methods, such as tensor voting and the structure tensor, since tensor-valued images can be seen as orientation scores with a low angular resolution (i.e., in case of 2-tensors $m_o \in \{-2, 0, 2\}$. We applied the results from Chapter 3 on tensor voting, to derive *steerable tensor voting*. This efficient method (see Table 3.2 on page 56 for an overview of the algorithmic complexities) was applied to a medical image analysis problem: detection of Electrophysiology (EP) catheters, showing that tensor voting leads to an increased noise robustness and helps to extract longer and smoother contours.

The last two chapters showed that $SE(2)$-convolutions are the natural way to linearly filter orientation scores and tensor images, for example to reduce noise and enhance elongated structures. $SE(2)$-convolutions can also be applied to operationalize differential operators on orientation scores. Therefore, in the next chapter we will discuss how to calculate regularized derivatives and how to apply them to obtain descriptive local features.

A line is a length without breadth.

Euclid (fl. 300 BC)

5 Regularized Derivatives and Local Features

This chapter is partly based on:

E. M. Franken, R. Duits, and B. M. ter Haar Romeny. Curvature estimation for enhancement of crossing curves. In *Proceedings of the 8th IEEE Computer Society Workshop on Mathematical Methods in Biomedical Image Analysis*, held in conjuction with the IEEE International Conference on Computer Vision, Rio de Janeiro, Brazil, October 14–20, 2007.

E.M. Franken and R. Duits. Crossing-preserving coherence-enhancing diffusion on invertible orientation scores. Accepted for publication in the *International Journal of Computer Vision (IJCV).*

5.1 Introduction

It is well known that taking derivatives of images is an ill-posed problem, which is made well-posed by adding regularization [46], e.g. by using Gaussian derivatives. Regularized derivatives are useful to obtain information on the local structure in images, e.g. edges, ridges, corners, and so on, see [66, Chapter 6] for an overview. In this chapter, we aim to do the same, i.e. computing regularized derivatives of orientation scores to estimate useful local features.

Once we have operationalized regularized derivatives in orientation scores, we will use them to measure useful local features in orientation scores. First we will discuss what kind of orientation score one should use as input for feature estimation. Then, we will consider three local features (see Figure 5.1) at each position $g \in SE(2)$ that are of interest especially for adaptive processing of orientation scores as described in the next chapter. The two features *curvature* $g \mapsto \kappa(g)$ and *deviation from horizontality* $g \mapsto d_{\mathrm{H}}(g)$ were already introduced in Subsection 2.8.7 on page 39 for curves in $SE(2)$. These two features can be calculated if one knows the tangent vector $\boldsymbol{c}(g)$ at each position g that locally fits best to the data. We will derive a method to estimate this tangent vector using the Hessian. The principle is analogous to orientation estimation using the Hessian on an image.

The third feature that we will introduce is *orientation confidence* $g \mapsto s(g)$, which is a scalar number indicating the confidence that an oriented structures is present at position g. This means that a low value indicates that the local neighborhood is isotropic, and a high value indicates that the neighborhood is anisotropic.

The chapter will end with an evaluation of the quality of the estimation of curvature and deviation from horizontality for different algorithms and different parameter values.

5.1.1 Related Work

The estimation of local curvature of elongated structures in images is a commonly studied problem in image processing. A well-known approach is *isophote curvature*. However, isophotes in images usually do not coincide with curves in images due to noise and blurring. Therefore, Duits [33, Section 5.5] derived a similar expression that can be used to estimate curvature of flowlines in vector fields, where the vector field can be obtained in a way that is optimal for the application at hand. Besides these approaches, several other methods have been proposed for robust curvature estimation of elongated structures in images [8, 106, 141, 59, 148]. All these methods can only estimate a single curvature at a single spatial position. The approach by Van Ginkel [137, 138] circumvents this problem by construct-

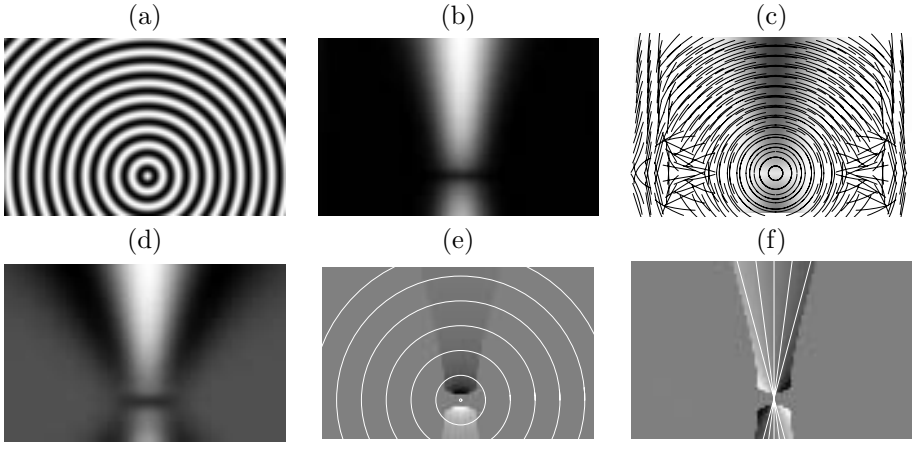


Figure 5.1: Example of feature estimation in an orientation score. (a) Input image f. (b) $|U_{\tilde{f}}|$ cf. (5.6) displayed for $\theta = 0$. This orientation score is used as input for feature estimation. (c) Estimated tangent vectors at orientation $\theta = 0$. The tangent vectors are displayed as circular arcs to show the estimated curvature as well as the deviation from horizontality. Note that the orientation and curvature estimation is undefined in isotropic regions in the orientation score, since the features are not well-defined there. (d) Orientation confidence s cf. (5.23) at $\theta = 0$. (e) Illustration of curvature κ, where the curvature values are only indicated by the grayvalue in the region where the orientation confidence is above a certain threshold, since outside of this region the values are irrelevant. Clearly, the displayed iso-κ-contours are situated on the superimposed circular arcs. (f) Deviation from horizontality d_{H}, where the deviation from horizontality values are again only indicated in the region where orientation confidence is above a certain threshold. Clearly, the displayed iso-d_{H}-contours, are orthogonal to the concentric circles in the image, and the vertical line is the $d_{\mathrm{H}} = 0$ line.

ing an orientation score. In the orientation score, the structure tensor is used to estimate the slope of a curve relative to the spatial plane, which is equal to the curvature. Our approach is closely related and inspired by Van Ginkel's approach, but it uses the Hessian instead of the structure tensor, and, as we will show in the evaluation, it does not suffer from biased estimates for higher curvature values.

5.2 Regularized Derivatives in Orientation Scores

We want to find useful local features using well-posed *left-invariant* derivative operators. We therefore aim to operationalize $\mathcal{D}(e^{t\mathcal{A}}U)$ where $\mathcal{D}$ is any derivative constructed from $\{\partial_\xi, \partial_\eta, \partial_\theta\}$, $e^{t\mathcal{A}}$ is the left-invariant diffusion operator described in Subsection 2.9.1, and U is an orientation score. Note that due to the noncommutative derivatives, the order of the regularization operator and differential op-

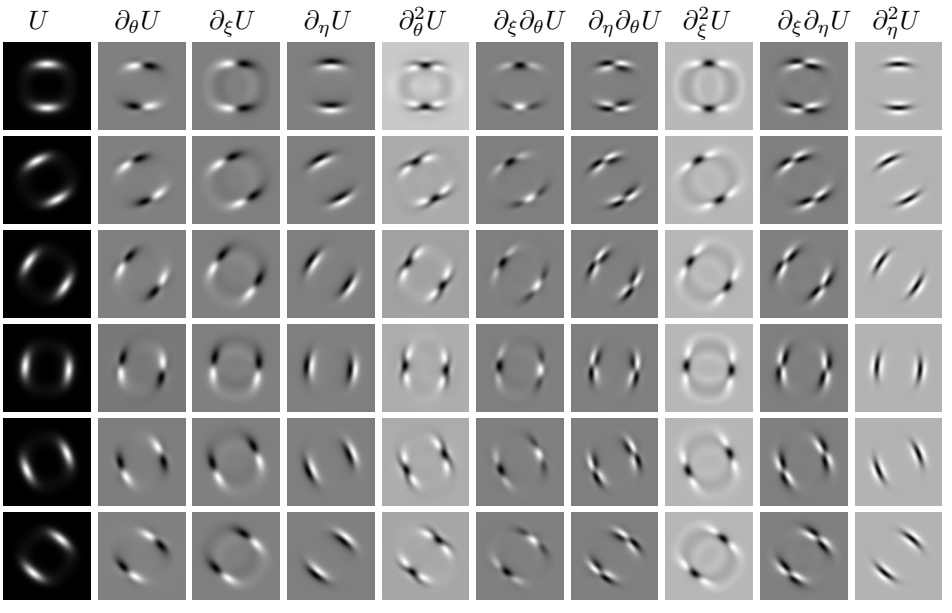


Figure 5.2: Example of a left-invariant Gaussian derivative jet of an orientation score U of an image consisting of a circle, as displayed in Figure 1.4 on page 6. Each row shows the same derivative for a different orientation.

erator matters, i.e. the diffusion should come first to ensure that the subsequent derivative operators are well-posed. We will first consider the use of diffusion with a diagonal diffusion tensor as regularizer. Subsequently, we will consider the special case μ-isotropic diffusion as regularizer, for which we can use normal Gaussian derivative implementations.

5.2.1 General Regularized Derivatives in Orientation Scores

We want to use anisotropic diagonal diffusion as regularizer to calculate left-invariant derivatives in $SE(2)$, i.e. diffusions satisfying the equation

$$\begin{cases} \partial_t W(g,t) = (D_{\xi\xi}\partial_\xi^2 + D_{\eta\eta}\partial_\eta^2 + D_{\theta\theta}\partial_\theta^2)W(g,t), \quad g \in SE(2), \quad t \geq 0, \\ W(g, t=0) = U(g), \end{cases} \tag{5.1}$$

Note that we do not consider diffusion tensors with nonzero off-diagonal components, i.e. curved or non-horizontal regularization kernels are not considered for the purpose of regularized derivatives. After all, our *aim* is to *use* the regularized derivatives to estimate curvature and deviation from horizontality, so including such prior knowledge is not sensible.

For a proper regularization in all directions we require $D_{\theta\theta} \neq 0$ and $D_{\xi\xi} \neq 0$. We do, however, not need to require $D_{\eta\eta} \neq 0$ since $D_{\eta\eta} = 0$ does render a smooth Green's function: the nonzero commutators diffusion in ξ-direction and θ-direction *induce* smoothing in the η-direction. This is the case since the Hörmander condition is also satisfied for $D_{\eta\eta} = 0$, see [39, Section 5.3].

We can implement regularized derivatives using (5.1) in one of the following ways

1. Sample the required derivative using an (approximate) analytical expression of the Green's function. Then, apply an $SE(2)$-convolution using one of the algorithms described in Chapter 3.

2. Use an iterative numerical scheme to diffuse the orientation score, e.g. one of the schemes that will be described in Section 6.4. Subsequently, numerically take the derivative, e.g. via the discrete Fourier transform using the property $\mathcal{F}[\partial_x a](\omega_x) = -i\omega_x \mathcal{F}[a](\omega_x)$, expressed for a 1D function a.

3. Use the same numerical scheme on a sampled spike $\delta_x \delta_y \delta_\theta$, with a very small amount of Gaussian blurring added to imrpove numerical stability, to obtain an approximation of the Greens's function. Then, either first numerically calculate the required derivative on the approximated Green's function and then apply an $SE(2)$-convolution, or first apply an $SE(2)$-convolution and then numerically calculate the required derivative on the result of the convolution.

For method 1, we need analytic expressions of the Green's function or *heat kernel* K. Both exact and approximate Green's functions have been derived for many cases of the diffusion kernels [39]. However, the exact Green's functions are not practical to use, since they consist of complicated special functions (i.e., *Mathieu functions*) that are expensive and complicated to calculate. Two approximations have been proposed that are easier to calculate: the Heisenberg approximation [39, Section 5.2] and the Gaussian estimates [39, Section 5.4]. However, both approximations are only applicable for the special case $D_{\eta\eta} = 0$, which is the case that was considered by Citti and Sarti [23] for the purpose of curve enhancement. However, we consider the freedom of selection of a diffusion scale in the η-direction very important as it is related to the selection of scale of the intensity profile orthogonal to the structures of interest. Furthermore, the first approximation only yields a simple expression for the resolvent case while for the time-process one still needs to solve an integral numerically. The second approximation, on the other hand, renders a non-differentiable approximation for the Green's function and is therefore not usable for the purpose of regularized derivatives at all.

Because of the practical problems with method 1, we reside to method 2 or method 3. Method 2 will most likely be slower especially if the effective size of the iteratively calculated Green's function is much smaller than the orientation

score. On the other hand, it is more accurate as one circumvents the interpolations involved in the $SE(2)$-convolution and the selection of the bounding box for the Green's function. The selection of the right method should be considered for each practical case separately.

5.2.2 Gaussian Derivatives in Orientation Scores

The implementations in the previous section are rather complicated and not exact. However, the implementation becomes much simpler if we restrict to the case of μ-isotropic diffusion cf. (2.84) on page 42, i.e.

$$\partial_t U = \left(\left(\partial_\xi^2 + \partial_\eta^2\right) + \mu^2 \partial_\theta^2\right) U = \left(\left(\partial_x^2 + \partial_y^2\right) + \mu^2 \partial_\theta^2\right) U. \tag{5.2}$$

Since this diffusion equation can be described using the commutative ∂_x, ∂_y, and ∂_θ derivatives, this equation simplifies to the diffusion equation in $\mathbb{R}^3$, except for the 2π-periodicity of the θ dimension. Therefore, the Green's function is approximated by the Gaussian kernel

$$G_{t_s,t_o}(x,y,\theta) = \frac{1}{8\sqrt{\pi^3 t_s^2 t_o}} e^{-\frac{x^2+y^2}{4t_s} - \frac{\theta^2}{4t_o}}, \quad \text{with } t_s = t \text{ and } t_o = \mu^2 t. \tag{5.3}$$

In this special case we can use standard separable implementations of Gaussian derivatives, but we have to be careful because of the non-commuting operators. A normal (i,j,k)th order isotropic Gaussian derivative implementation for a 3D image f adheres to the right-hand side of the following equation

$$\partial_x^i \partial_y^j \partial_z^k e^{t(\partial_x^2+\partial_y^2+\partial_z^2)} f = \partial_x^i e^{t\,\partial_x^2} \left(\partial_y^j e^{t\,\partial_y^2} \left(\partial_z^k e^{t\,\partial_z^2} f\right)\right). \tag{5.4}$$

The right-hand side of this equation shows that the operator is *separable*, since it consists of a concatenation of three operators that are each one-dimensional only. When using a direct convolution implementation the complexity drops from $\mathcal{O}\left(N^3K^3\right)$ (3D convolution) to $\mathcal{O}\left(N^3K\right)$ ($3N^2$ 1D convolutions) where N is the image size in one dimension and K the size of the one-dimensional sampled Gaussian derivative.

We want to use the same implementations to construct Gaussian derivatives in orientation scores, meaning that we have to ensure that the same permutation of differential operators and regularization operators is allowed. By noting that

$$\begin{aligned} \partial_\xi^i\, \partial_\eta^j\, \partial_\theta^k\, e^{t_o\partial_\theta^2 + t_s(\partial_\xi^2+\partial_\eta^2)} &= \partial_\xi^i \partial_\eta^j\, e^{t_s(\partial_x^2+\partial_y^2)}\, \partial_\theta^k\, e^{t_o\partial_\theta^2}, \\ \text{and } \partial_\theta^k\, \partial_\xi^i\, \partial_\eta^j\, e^{t_o\partial_\theta^2 + t_s(\partial_\xi^2+\partial_\eta^2)} &\neq \partial_\theta^k\, e^{t_o\partial_\theta^2}\, \partial_\xi^i\, \partial_\eta^j\, e^{t_s(\partial_x^2+\partial_y^2)}, \end{aligned} \tag{5.5}$$

we conclude that we should first calculate the orientational derivative ∂_θ and then the commuting spatial derivatives $\{\partial_\xi, \partial_\eta\}$, which are calculated from the Cartesian derivatives $\{\partial_x, \partial_y\}$ using $\partial_\xi = \cos\theta\,\partial_x + \sin\theta\,\partial_y$ and $\partial_\eta = -\sin\theta\,\partial_x + \cos\theta\,\partial_y$.

The commutator relations $[\partial_\theta, \partial_\xi] = \partial_\eta$ and $[\partial_\theta, \partial_\eta] = -\partial_\xi$ (introduced in Subsection 2.8.2) allow to rewrite the derivatives in this "canonical" order. For instance, the derivative $\partial_\xi \partial_\theta$ can be calculated directly with Gaussian derivatives, while $\partial_\theta \partial_\xi$ must be operationalized with Gaussian derivatives as $\partial_\xi \partial_\theta + \partial_\eta$. Figure 5.2 displays the first and second order canonically ordered Gaussian derivative jet of an orientation score.

In the rest of this thesis, we will use Gaussian derivatives for regularized left-invariant derivatives in orientation scores. One can argue that choosing $D_{\xi\xi} > D_{\eta\eta}$ is a better choice, because such regularization is better adapted to the local anisotropic structure of the orientation score, which is mainly aligned with the ξ-direction. On the other hand, in an orientation score obtained from a noisy image, most noise can be found in the η-direction. This is caused by the strongly oriented shape of the orientation score construction kernels of Subsection 2.4.1 on page 19, which already leads to blurring in the direction of the structure and not orthogonal to it. These two opposing considerations suggest that using $D_{\xi\xi} = D_{\eta\eta}$ is a good practical compromise, which is also the best choice concerning computation speed.

5.3 Tangent Vector Estimation

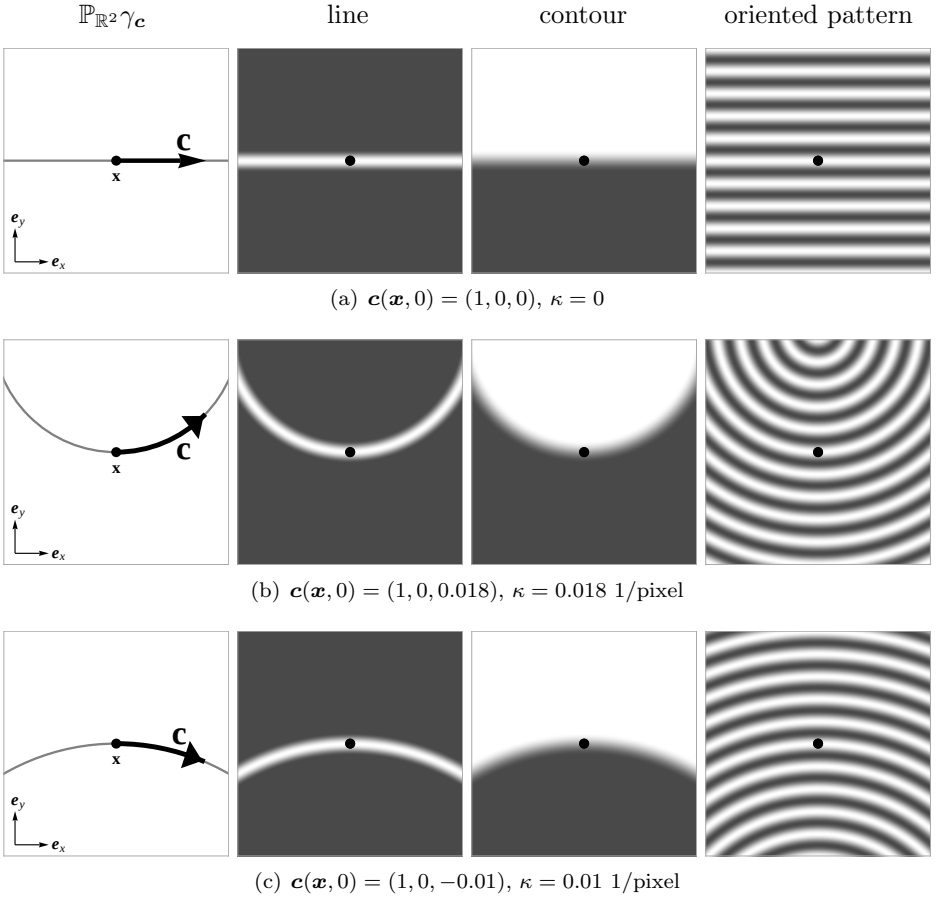
For each position $g \in SE(2)$ in the orientation score, we aim to find the exponential curve $g\,\gamma_{\boldsymbol{c}}$, see Subsection 2.8.6, that locally first best to the data. Figure 5.3 illustrates this on the image plane $\mathbb{R}^2$. The local optimal fitting exponential curve should not only be feasibly estimated at the centerlines of curves, but also if we shift a bit away from the centerline. At non-oriented positions $g \in SE(2)$, however, the tangent vector $\boldsymbol{c}(g)$ is not defined.

We aim to handle lines, contours, and oriented patterns on the same footing, i.e. for all cases we want to obtain an appropriate tangent vector estimate, as shown in Figure 5.3. Remember that in Subsection 2.4.1 we described the design of an invertible orientation score, which had to approximately fulfill the quadrature property in η-direction, i.e. orthogonal to horizontal oriented structures. Therefore the input to our tangent vector estimation procedure is the *phase invariant* orientation score (see Figure 5.1b), which is found by

$$\check{U} = \left| U_{\tilde{f}} \right|, \quad \text{with } \tilde{f} = f - (f *_{\mathbb{R}^2} G_{\sigma_s}). \tag{5.6}$$

By construction of $\tilde{f}$ from the original image f, the low frequency components are left out, which ensures that we have *mean grey-value invariance*. Figure 5.1b

If we follow an oriented structure in the orientation score $\check{U}$, the left-invariant gradient $\nabla \check{U} = (\partial_\xi, \partial_\eta, \partial_\theta)^{\mathrm{T}} \check{U}$ at all positions should remain constant. For example on the centerline of a curve the gradient remains zero, while the gradient will have



(a) $\boldsymbol{c}(\boldsymbol{x}, 0) = (1, 0, 0)$, $\kappa = 0$

(b) $\boldsymbol{c}(\boldsymbol{x}, 0) = (1, 0, 0.018)$, $\kappa = 0.018$ 1/pixel

(c) $\boldsymbol{c}(\boldsymbol{x}, 0) = (1, 0, -0.01)$, $\kappa = 0.01$ 1/pixel

Figure 5.3: Illustration of the interpretation of a tangent vector $\boldsymbol{c}(\boldsymbol{x}, 0) \in T_{(\boldsymbol{x},0)}(SE(2))$ projected onto the image plane $\mathbb{R}^2$. The first column shows three examples of exponential curves $\gamma_{\boldsymbol{c}}$ in $SE(2)$ projected onto $\mathbb{R}^2$ and the tangent vector at position $(\boldsymbol{x}, 0)$. For nonzero curvature cf. (b) and (c) the tangent vector $\boldsymbol{c}$ is visualized as a "curved tangent vector". At the center position, indicated by the black dot, the projected exponential curves shown in the first column have the same orientation and curvature as the elongated structures in the image patches shown in the second, third, and fourth column. This means that in the corresponding orientation score $\boldsymbol{c}$ is tangent to the local elongated structure. The image patches have a size of 100×100 pixels. The aim of tangent vector estimation is to automatically obtain the tangent vector $\boldsymbol{c}(\boldsymbol{x}, \theta)$ from the orientation score corresponding to the image.

a small constant η-component if we are a little bit off from the centerline. Thus, we formulate a minimization problem that minimizes over the "iso-contours" of the left-invariant gradient vector at position g, leading to

$$\boldsymbol{c}^*(g) = \arg\min_{\boldsymbol{c}(g)} \left\{ \left\| \frac{\mathrm{d}}{\mathrm{d}t}(\nabla \check{U}(g\,\gamma_{\boldsymbol{c}(g)}(t))) \Big|_{t=0} \right\|_\mu^2 \Big| \, \|\boldsymbol{c}(g)\|_\mu = 1 \right\}, \tag{5.7}$$

where $\boldsymbol{c}(g) = \big(c^\xi(g), c^\eta(g), c^\theta(g)\big)$, the norm $\|\cdot\|_\mu$ is defined in Subsection 2.8.4 on page 36, and $g\,\gamma_{\boldsymbol{c}g}(t) = g\exp\big(t(c^\xi\partial_x + c^\eta\partial_y + c^\theta\partial_\theta)\big)$, recall (2.72).

The minimizing equation in (5.7) is a norm of a covector and can be rewritten as

$$\left\| \frac{\mathrm{d}}{\mathrm{d}t}(\nabla \check{U}(\gamma_{g\,\boldsymbol{c}(g)}(t))) \Big|_{t=0} \right\|_\mu^2 = \|\nabla(\nabla \check{U}(g))\,\dot{\gamma}_{\boldsymbol{c}(g)}(0)\|_\mu^2 = \big\|\mathcal{H}\check{U}(g)\boldsymbol{c}(g)\big\|_\mu^2. \tag{5.8}$$

For notational simplicity, from now on we do not explicitly denote the orientation score position g, and we rewrite the right-hand side of the latter equation as

$$\begin{aligned} \big\|\mathcal{H}\check{U}\boldsymbol{c}\big\|_\mu^2 = \big\|\mathcal{H}\check{U}(g)\boldsymbol{c}\big\|_\mu^2 = (\mathcal{H}\check{U}\boldsymbol{c}, \mathcal{H}\check{U}\boldsymbol{c})_\mu = (\mathbf{M}_\mu\mathcal{H}\check{U}\boldsymbol{c}, \mathbf{M}_\mu\mathcal{H}\check{U}\boldsymbol{c})_1 \\ = (\boldsymbol{c}, (\mathcal{H}\check{U})^{\mathrm{T}}\mathbf{M}_\mu^2(\mathcal{H}\check{U})\boldsymbol{c})_1, \end{aligned} \tag{5.9}$$

where $\mathbf{M}_\mu = \mathrm{diag}\{1/\mu, 1/\mu, 1\}$ and $(\cdot,\cdot)_1$ denotes the normal $\mathbb{R}^3$ inner product. The Hessian $\mathcal{H}\check{U}$ is defined by

$$\begin{aligned} \mathcal{H}\check{U} = \nabla(\nabla\check{U}) &= \begin{pmatrix} \partial_\xi^2\check{U} & \partial_\eta\partial_\xi\check{U} & \partial_\theta\partial_\xi\check{U} \\ \partial_\xi\partial_\eta\check{U} & \partial_\eta^2\check{U} & \partial_\theta\partial_\eta\check{U} \\ \partial_\xi\partial_\theta\check{U} & \partial_\eta\partial_\theta\check{U} & \partial_\theta^2\check{U} \end{pmatrix} \\ &= \begin{pmatrix} \partial_\xi^2\check{U} & \partial_\xi\partial_\eta\check{U} & \partial_\xi\partial_\theta\check{U} + \partial_\eta\check{U} \\ \partial_\xi\partial_\eta\check{U} & \partial_\eta^2\check{U} & \partial_\eta\partial_\theta\check{U} - \partial_\xi\check{U} \\ \partial_\xi\partial_\theta\check{U} & \partial_\eta\partial_\theta\check{U} & \partial_\theta^2\check{U} \end{pmatrix}. \end{aligned} \tag{5.10}$$

The side condition $\|\boldsymbol{c}\|_\mu = 1$ can be rewritten as

$$\|\boldsymbol{c}\|_\mu = (\mathbf{M}_\mu^{-1}\boldsymbol{c}, \mathbf{M}_\mu^{-1}\boldsymbol{c})_1 = (\boldsymbol{c}, \mathbf{M}_\mu^{-2}\boldsymbol{c})_1. \tag{5.11}$$

By Euler-Lagrange minimization $(\nabla_{\boldsymbol{c}}\|(\mathcal{H}\check{U})\boldsymbol{c}\|_\mu^2 - \lambda(1 - \|\boldsymbol{c}\|_\mu) = 0$ we get for the optimum $\boldsymbol{c}^*$:

$$(\mathcal{H}\check{U})^{\mathrm{T}}\mathbf{M}_\mu^2(\mathcal{H}\check{U})\boldsymbol{c}^* = \lambda\mathbf{M}_\mu^{-2}\boldsymbol{c}^*. \tag{5.12}$$

This can be rewritten as

$$(\mathbf{M}_\mu\mathcal{H}\check{U}\mathbf{M}_\mu)^{\mathrm{T}}(\mathbf{M}_\mu\mathcal{H}\check{U}\mathbf{M}_\mu)\,\tilde{\boldsymbol{c}}^* = \lambda\,\tilde{\boldsymbol{c}}^*, \tag{5.13}$$

where $\tilde{\boldsymbol{c}}^* = \mathbf{M}_\mu^{-1}\boldsymbol{c}^*$ which amounts to eigensystem analysis of the symmetric 3×3 matrix $(\mathbf{M}_\mu\mathcal{H}\check{U}\mathbf{M}_\mu)^{\mathrm{T}}(\mathbf{M}_\mu\mathcal{H}\check{U}\mathbf{M}_\mu)$, where one of the three eigenvectors gives $\tilde{\boldsymbol{c}}^*$.

The eigenvector with the smallest corresponding eigenvalue is selected as tangent vector $\tilde{\boldsymbol{c}}^*$, and the desired tangent vector $\boldsymbol{c}^*$ is then given by $\boldsymbol{c}^* = \mathbf{M}_\mu \tilde{\boldsymbol{c}}^*$. The curvature and deviation from horizontality can now be obtained from $\boldsymbol{c}^*$ using the expression from Subsection 2.8.7 on page 39, i.e.

$$\kappa = \frac{c^\theta \operatorname{sign}(c^\xi)}{\sqrt{(c^\eta)^2 + (c^\xi)^2}}, \tag{5.14}$$

and

$$d_{\mathrm{H}} = \arctan\left(\frac{c^\eta}{c^\xi}\right). \tag{5.15}$$

Optionally, an increased noise-robustness can be achieved by component-wise blurring of the matrix $(\mathbf{M}_\mu \mathcal{H}\check{U}\mathbf{M}_\mu)^{\mathrm{T}}(\mathbf{M}_\mu \mathcal{H}\check{U}\mathbf{M}_\mu)$ before performing eigensystem analysis, i.e. (5.13) is replaced by

$$\left(G_{\rho_s,\rho_o} * (\mathbf{M}_\mu \mathcal{H}\check{U}\mathbf{M}_\mu)^{\mathrm{T}}(\mathbf{M}_\mu \mathcal{H}\check{U}\mathbf{M}_\mu)\right) \tilde{\boldsymbol{c}}^* = \lambda\, \tilde{\boldsymbol{c}}^*, \tag{5.16}$$

where ρ_s and ρ_o are the spatial and orientational scales respectively. The derivatives in $\mathcal{H}\check{U}$ are regularized derivatives, either using Gaussian derivatives cf. Subsection 5.2.2 or generalized derivatives cf. Subsection 5.2.1. The post-blurring ensures that matrices that describe the local structure inaccurately become more consistent with the surrounding. This approach is similar to the structure tensor where one applies post-blurring on the matrices formed by the dyadic product of the gradient with itself.

5.3.1 Enforcing Horizontality

The procedure described above will not work for periodic oriented patterns, e.g. the patterns shown in the right column of Figure 5.3. This is caused by the way we construct the input feature orientation score $\check{U}$ cf. (5.6). Regular oriented patterns in the image, such as the image in Figure 5.1a, result in a flat response in $\check{U}$, as is illustrated in Figure 5.1b. Consequently, tangent vectors $\boldsymbol{c}(g)$ tangent to oriented patterns at g are not well-defined.

For images with oriented patterns, the problem is solved by forcing the deviation from horizontality to zero[1]. On curves that are not exactly horizontal, for example because the number of sampled orientations N_o is chosen to be fairly low, the performance of the tangent vector estimation is expected to decrease.

Horizontality is imposed by forcing c^η to zero in (5.7). In the minimization term,

[1] Alternatively one could use a different (application-specific) feature orientation score $\check{U}$ for feature estimation which is better suitable for oriented patterns.

$(\mathcal{H}\check{U}\boldsymbol{c}^*)$ can now be rewritten as

$$\mathcal{H}\check{U}\boldsymbol{c}\big|_{c^\eta=0} = \mathcal{H}_{\mathrm{hor}}\check{U}\boldsymbol{c}_{\mathrm{hor}} = \begin{pmatrix} \partial_\xi\partial_\eta\check{U} & \partial_\theta\partial_\eta\check{U} \\ \partial_\xi\partial_\theta\check{U} & \partial_\theta^2\check{U} \\ \partial_\xi^2\check{U} & \partial_\theta\partial_\xi\check{U} \end{pmatrix} \begin{pmatrix} c^\xi \\ c^\theta \end{pmatrix}. \tag{5.17}$$

Now the Euler Lagrange equation gives

$$(\mathbf{M}_\mu\,\mathcal{H}_{\mathrm{hor}}\check{U}\,\mathbf{M}_{\mu,\mathrm{hor}})^{\mathrm{T}}(\mathbf{M}_\mu\,\mathcal{H}_{\mathrm{hor}}\check{U}\,\mathbf{M}_{\mu,\mathrm{hor}})\,\tilde{\boldsymbol{c}}^*_{\mathrm{hor}} = \lambda\,\tilde{\boldsymbol{c}}^*_{\mathrm{hor}}, \tag{5.18}$$

where $\tilde{\boldsymbol{c}}^*_{\mathrm{hor}} = \mathbf{M}_{\mu,\mathrm{hor}}\boldsymbol{c}^*_{\mathrm{hor}}$ and $\mathbf{M}_{\mu,\mathrm{hor}} = \mathrm{diag}\{1/\mu, 1\}$. If one applies post-blurring this equation becomes

$$\big(G_{\rho_s,\rho_o} * (\mathbf{M}_\mu\,\mathcal{H}_{\mathrm{hor}}\check{U}\,\mathbf{M}_{\mu,\mathrm{hor}})^{\mathrm{T}}(\mathbf{M}_\mu\,\mathcal{H}_{\mathrm{hor}}\check{U}\,\mathbf{M}_{\mu,\mathrm{hor}})\big)\;\tilde{\boldsymbol{c}}^*_{\mathrm{hor}} = \lambda\,\tilde{\boldsymbol{c}}^*_{\mathrm{hor}}. \tag{5.19}$$

Solving equation (5.18) or (5.19) amounts to eigensystem analysis of a symmetric 2×2 matrix. The eigenvector corresponding to the smallest eigenvalue should be selected and the curvature is given by

$$\kappa = \frac{c^\theta}{c^\xi}. \tag{5.20}$$

The deviation from horizontality is inherently zero in this case. The fact that we have 2×2 matrices instead of 3×3 is a computational advantage of this approach.

5.3.2 Structure Tensor Approach

An alternative approach to the tangent vector estimation using the Hessian is to use the structure tensor as has been proposed by Van Ginkel [137, 138] for the purpose of curvature estimation. In this approach one simply replaces the Hessian by the structure tensor, defined by

$$\mathcal{S}\check{U}(\boldsymbol{x},\theta) = \tilde{\mathbf{R}}_\theta \cdot \left(G_{\rho_s,\rho_o} * \left(\begin{pmatrix} \partial_x\check{U} \\ \partial_y\check{U} \\ \partial_\theta\check{U} \end{pmatrix} \cdot \begin{pmatrix} \partial_x\check{U} \\ \partial_y\check{U} \\ \partial_\theta\check{U} \end{pmatrix}^{\mathrm{T}} \right) \right)(\boldsymbol{x},\theta) \cdot \tilde{\mathbf{R}}_\theta^{\mathrm{T}}, \tag{5.21}$$

with

$$\tilde{\mathbf{R}}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.22}$$

where the derivatives, which are not the left-invariant ones, are implemented by Gaussian derivatives, and G_{ρ_s,ρ_o} denotes the Gaussian smoothing kernel that is applied componentwise to the structure tensor. On the resulting structure tensors we apply eigensystem analysis in exactly the same manner as described above for $(\tilde{\mathcal{H}}\check{U})^{\mathrm{T}}(\tilde{\mathcal{H}}\check{U})$. In Subsection 5.5.1 we will see the advantages and disadvantages of the structure tensor approach.

5.4 Orientation Confidence

In a 3D image, an often used measure for *orientation confidence* is the sum of the two second order derivatives orthogonal to the orientation of the elongated structure, which amounts to the sum of the two largest absolute eigenvalues of the Hessian matrix. In fact, this corresponds to the Laplacian in the plane orthogonal to the line structure.

In the orientation score we adopt the same approach. As measure for orientation confidence s we take the Laplacian in the plane orthogonal to the line, which is calculated at $g \in SE(2)$ by

$$s(g) = -\Delta_{\circ}\check{U}(g) = -((\boldsymbol{e}_1^{\circ}(g))^{\mathrm{T}}\mathcal{H}\check{U}(g)\boldsymbol{e}_1^{\circ}(g) + (\boldsymbol{e}_2^{\circ}(g))^{\mathrm{T}}\mathcal{H}\check{U}(g)\boldsymbol{e}_2^{\circ}(g)), \tag{5.23}$$

where $\boldsymbol{e}_1^{\circ}$ and $\boldsymbol{e}_2^{\circ}$ are two vectors that are orthonormal to the tangent vector $\boldsymbol{c}$ with respect to the inner product defined in equation (2.65), i.e.

$$(\boldsymbol{c}, \boldsymbol{e}_1^{\circ})_{\mu} = 0, \quad (\boldsymbol{c}, \boldsymbol{e}_2^{\circ})_{\mu} = 0, \quad (\boldsymbol{e}_i^{\circ}, \boldsymbol{e}_j^{\circ})_{\mu} = \delta_{ij},\ i, j \in \{1, 2\}. \tag{5.24}$$

The minus sign in (5.23) is included in order to get a positive response for oriented structures: an oriented structure at position g always renders a convex hill in the intensity landscape around $\check{U}(g)$, no matter whether the structure is dark or light relative to the background. Therefore an oriented structure always yields a negative value for the second order derivative.

5.5 Results

For the evaluation of the quality of the estimations of curvature and deviation from horizontality, we use the test images displayed in Figure 5.4, which all have a resolution of 128×128 pixels. We evaluate the method on test images with both oriented patterns and lines. We do not include test images with contours, since there is no reason to assume different results for lines and contours due to the construction of the feature orientation score cf. (5.6). The test image "circle pattern" is chosen because it contains a wide range of different curvatures, and because it is a useful test image for evaluating the quality of the method on regular oriented patterns. Test image set "separate circles" in Figure 5.4 represents a collection of test images containing circles with radii $\{5, 6, 7, \ldots, 50\}$. In this way we can evaluate curvature estimation for a wide range of curvatures on separate line structures in images.

To obtain noisy images, Gaussian noise is added. The following definition is used for the signal-to-noise ratio (SNR)

$$\mathrm{SNR} = \frac{\max(\mathrm{image}) - \min(\mathrm{image})}{2\sigma_{\mathrm{noise}}^2}, \tag{5.25}$$

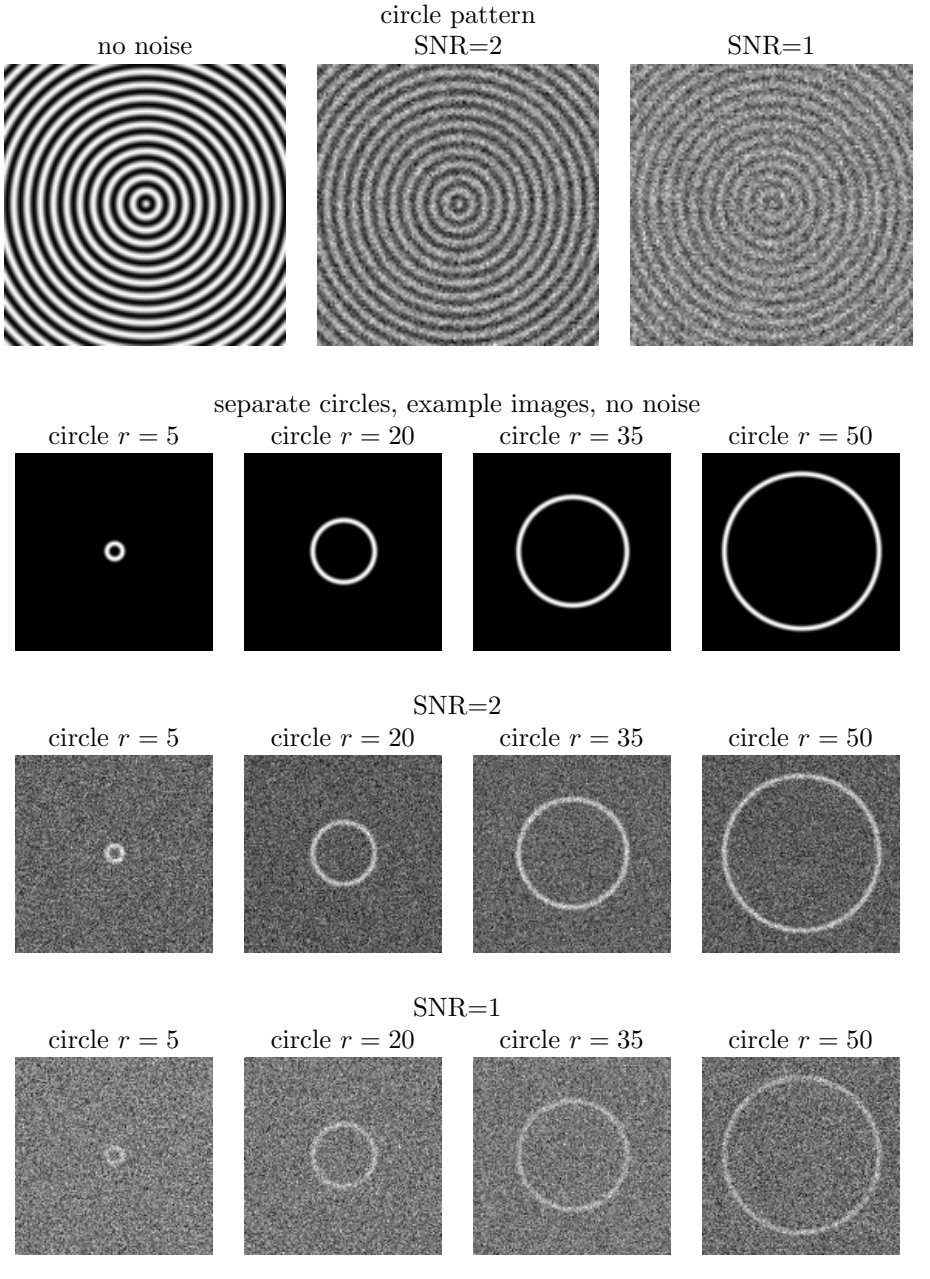


Figure 5.4: Curvature estimation test images. All images are 128×128 pixels. The test images "separate circles" represent a collection of test images containing circles with radii $\{5, 6, 7, \ldots, 50\}$.

where σ_{noise} is the standard deviation of the Gaussian noise.

In all experiments, the images are transformed to orientation scores using the orientation score transform of Subsection 2.4.1 on page 19 with parameter values $s_\theta = \pi/32$, $k = 2$, $q = 8$, $t = 660$, and $\sigma_s = 50$. Subsequently, unless denoted differently, we use as parameters for the Hessian method $t_s = 9.5$, $\rho_s = 0.5$, and $\mu = 0.08$.

For the test images (Figure 5.4) we do know the ground truth orientation $\boldsymbol{x} \mapsto \theta_{\text{true}}(\boldsymbol{x})$ and curvature $\boldsymbol{x} \mapsto \kappa_{\text{true}}(\boldsymbol{x})$. Furthermore, for all images a mask is defined specifying at which position the curvature should be taken into account. For "circle pattern" this ensures that all circles with radii 5 to 50 are taken into account. This range is chosen such that the results are not influenced too much by boundary artefacts and no problems arise due to the singularity in the middle of the image (where the curvature is theoretically infinite). For "separate circles", each image with a different circle radius has its own mask, ensuring that only curvature estimates are taken into account at pixel positions that have a distance to the centerline of at most 1 pixel.

To display the results, we take for each spatial position which is in the mask the curvature estimate κ_{est} for the known orientation and compare it to the ground truth curvature. In all experiments we display a density plot showing $|1/\kappa_{\text{est}}|$ (vertical) against the true curvature $|1/\kappa_{\text{true}}|$ (horizontal). For "separate circles", the curvature estimation results for all radii are fused into one density plot.

In the forthcoming subsections we will evaluate

1. Using the Hessian versus structure tensor approach;
2. The effect of using the full Hessian versus enforcing horizontality;
3. Varying the numbers of orientations;
4. The effect of crossing curves.

5.5.1 Hessian versus Structure Tensor

The parameters for the Hessian approach are given above. For the structure tensor approach (see Subsection 5.3.2 and [137]), we use parameters $t_s = 2.5$, $\rho_s = 7.5$, and $\mu = 0.08$. The parameters are chosen such that the total amount of Gaussian blurring is the same as the Hessian case, in order to take the same neighborhood into account. The derivative scale and post-blurring scale for both methods individually are chosen such that both methods seem to perform optimally.

The results are shown in Figure 5.5. We observe that

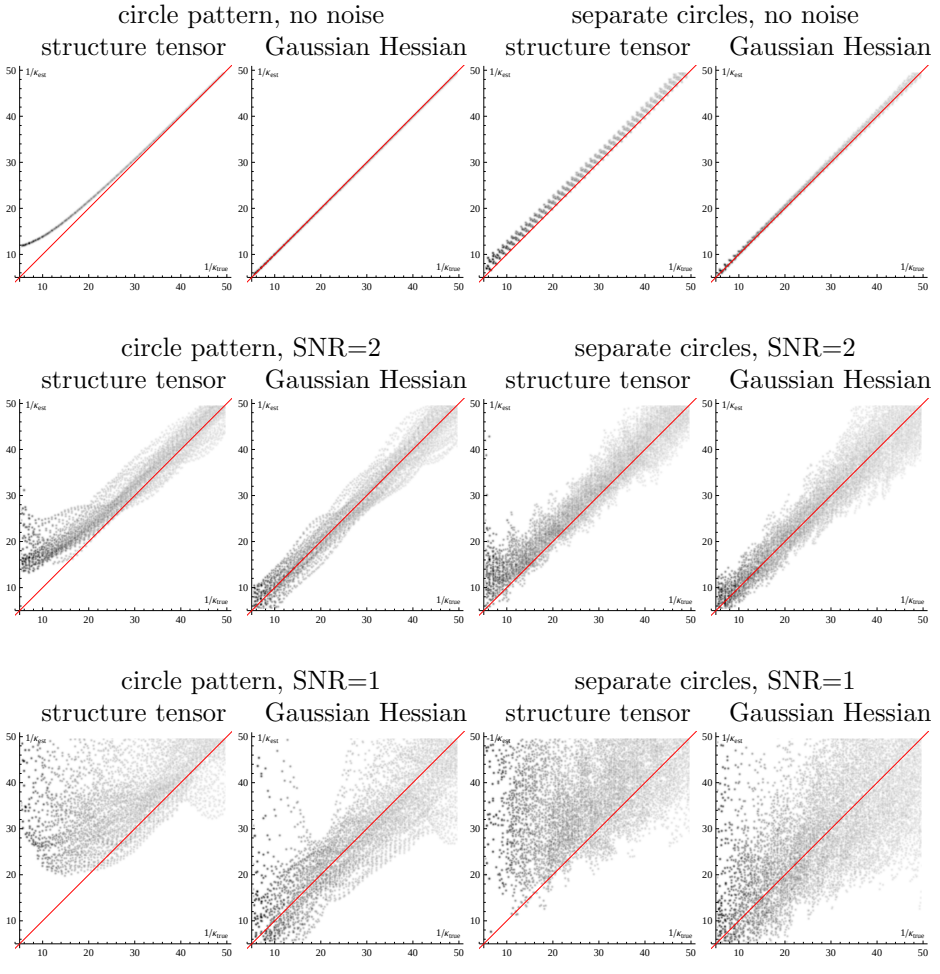


Figure 5.5: Curvature estimation results for structure tensor and gaussian hessian, with enforcing horizontality. See text (Subsection 5.5.1) for details.

- The results of the Gaussian Hessian method are quite good, although the performance decreases significantly as noise is added. Further tuning of the scale parameters might lead to more noise robustness.

- On "circle pattern", the structure tensor has a large bias: the curvature estimates are too small over a wide range of the highest curvature values. The post-blurring that is needed to obtain useful information for the structure tensor, causing neighboring circles with other curvature values to influence the estimate. The fact that the bias is negative is caused by the fact that in our image, the majority of the pixels around an arbitrary position $\boldsymbol{x}$ have true curvature values that are smaller than the curvature at $\boldsymbol{x}$.

- On "separate circles", the structure tensor shows a "zigzag edge". This is caused by the mask we use for all individual circle images, which is 3 pixels wide. Apparently, the structure tensor estimate is negatively biased at positions off from the centerline.

5.5.2 Effect of Enforcing Deviation from Horizontality

Figure 5.6 shows the difference between taking the full Hessian cf. (5.16) and enforcing horizontality cf. (5.19) for curvature estimates on horizontal and non-horizontal positions. A curvature estimate at a non-horizontal position $d_{\mathrm{H}} = \pi/4$ is obtained by taking the curvatures at $\theta_{\mathrm{true}} + \pi/4$. It is considered important to have a good curvature estimate also on slightly non-horizontal positions as well for the nonlinear diffusion processes in the next chapter.

We observe in Figure 5.6 that

- For the cases $d_{\mathrm{H}} = 0$, in the noise-free case it does not matter whether to enforce horizontality or to take the full Hessian.

- For the cases $d_{\mathrm{H}} = \pi/4$, we observe that the horizontal Gaussian Hessian estimates fail, while the results are reasonably good for the full Gaussian Hessian method. This holds for both the "circle pattern" and the "separate circles".

5.5.3 Estimation of Deviation from Horizontality

In this experiment we consider the full Gaussian Hessian method and study the quality of the estimation of curvature *and* deviation from horizontality, as function of noise and the number of orientations.

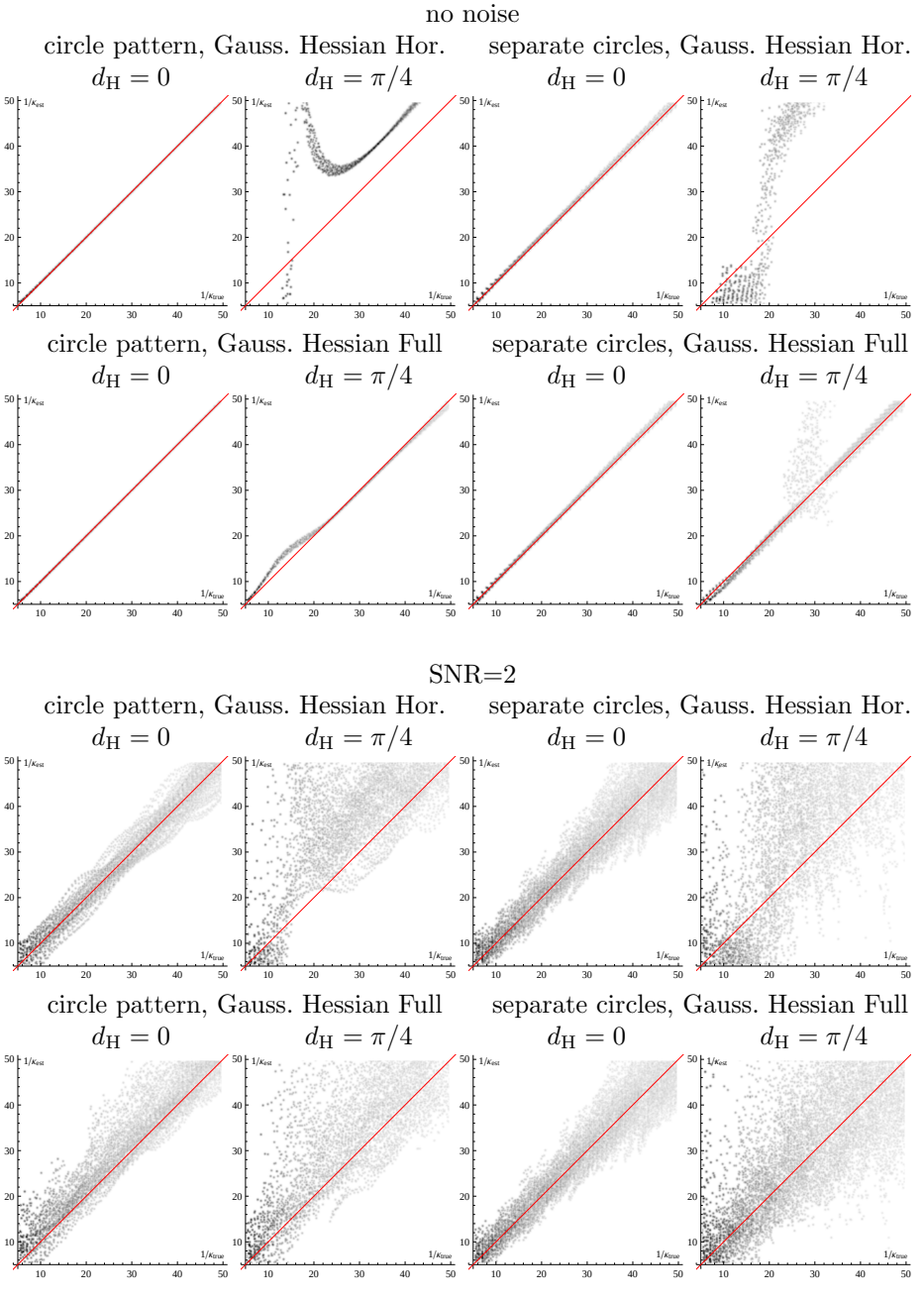


Figure 5.6: Curvature estimation results showing the difference between taking the full Hessian and enforcing horizontality on curvature estimates on horizontal and non-horizontal positions. See text (Subsection 5.5.2) for details.

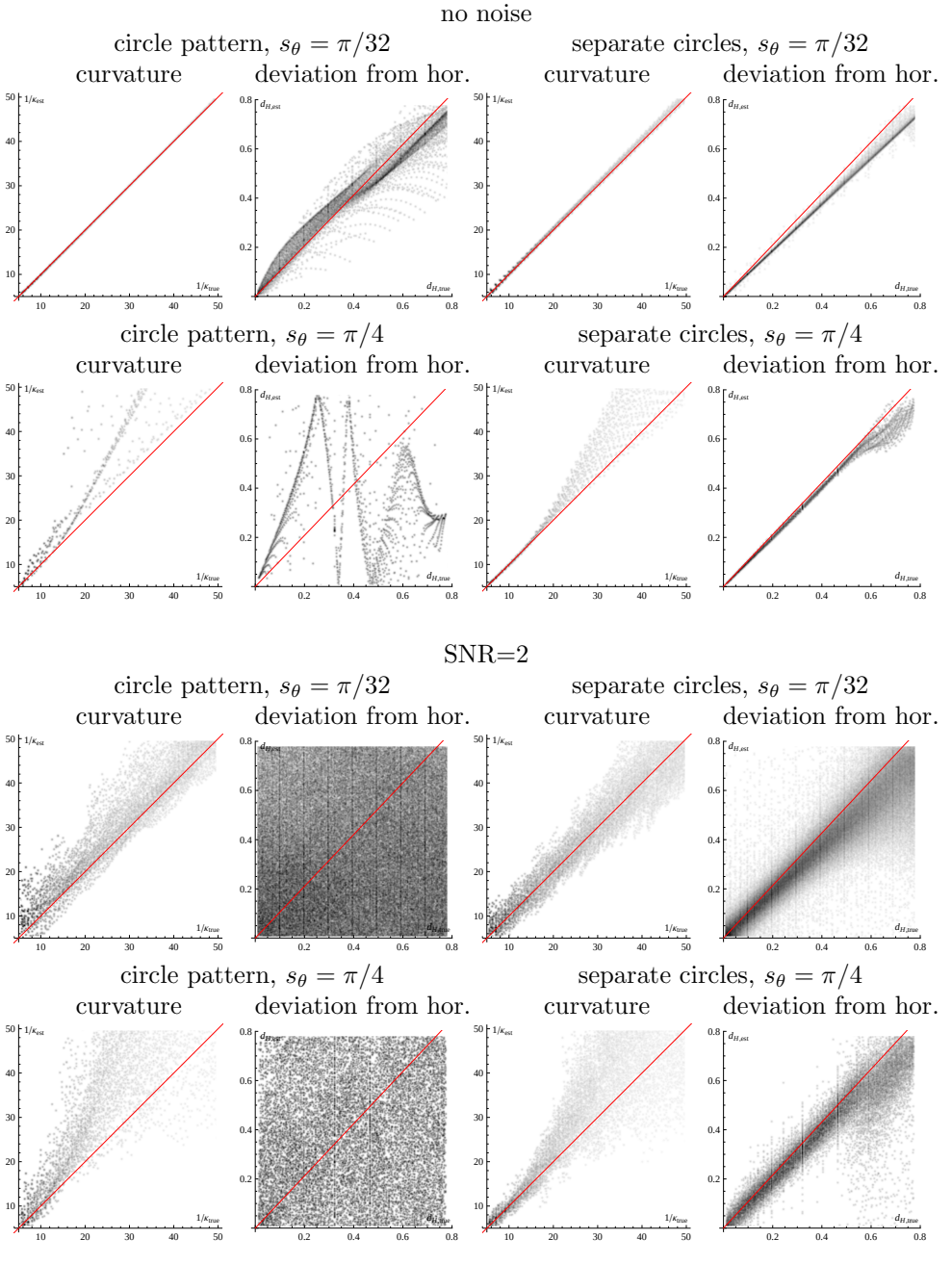


Figure 5.7: Results of curvature estimation and deviation from horizontality estimation for two different numbers of sampled orientations. See text (Subsection 5.5.3) for details.

For the evaluation of the deviation from horizontality, we display a density plot showing $|d_{\mathrm{H}_{\mathrm{est}}}|$ (vertical) cf. (5.15) against the true curvature $|d_{\mathrm{H}_{\mathrm{true}}}|$ (horizontal), where the true deviation from horizontality is obtained by $|d_{\mathrm{H}_{\mathrm{true}}}(\boldsymbol{x}, \theta)| = |\theta - \theta_{\mathrm{true}}(\boldsymbol{x})|$. The plots are shown for $0 \leq |d_{\mathrm{H}}| \leq \frac{\pi}{4}$ since this is the range over which we consider a correct estimation of the deviation from horizontality most important.

Figure 5.7 shows the results. The major points are

- The estimates for d_{H} on "circle pattern" are poor. For $s_\theta = \pi/4$ and for the noisy cases they are completely unreliable. This is as expected, as the reason for this was already discussed in Subsection 5.3.1, but it is important to take into account in practice.
- For "separate circles" the d_{H} estimates are quite good, although they are slightly negatively biased.

5.5.4 Handling Crossing Curves

Figure 5.8 shows the curvature estimation in the displayed image with crossing curves, for different noise levels and number of orientations. The most important findings are

- A decrease in performance can be seen compared to the "circle pattern" test image. After all, an orientation score transform cannot perfectly separate the responses of the different curves.
- The method is sensitive to a decrease in the number of orientations. The higher the number of orientations, the better the oriented structures are separated.

5.6 Conclusions

We have shown how one can compute left-invariant derivatives where left-invariant diffusion with a diagonal diffusion tensor acts as the regularizer. We showed that we can use normal Gaussian derivatives for the case $D_{\xi\xi} = D_{\eta\eta}$.

We have used the regularized derivatives to estimate a tangent vector, which is tangent to the locally best fitting exponential curve on each position in the orientation score. Using this tangent vector we have calculated three features describing the local structure in the orientation score: curvature κ, deviation

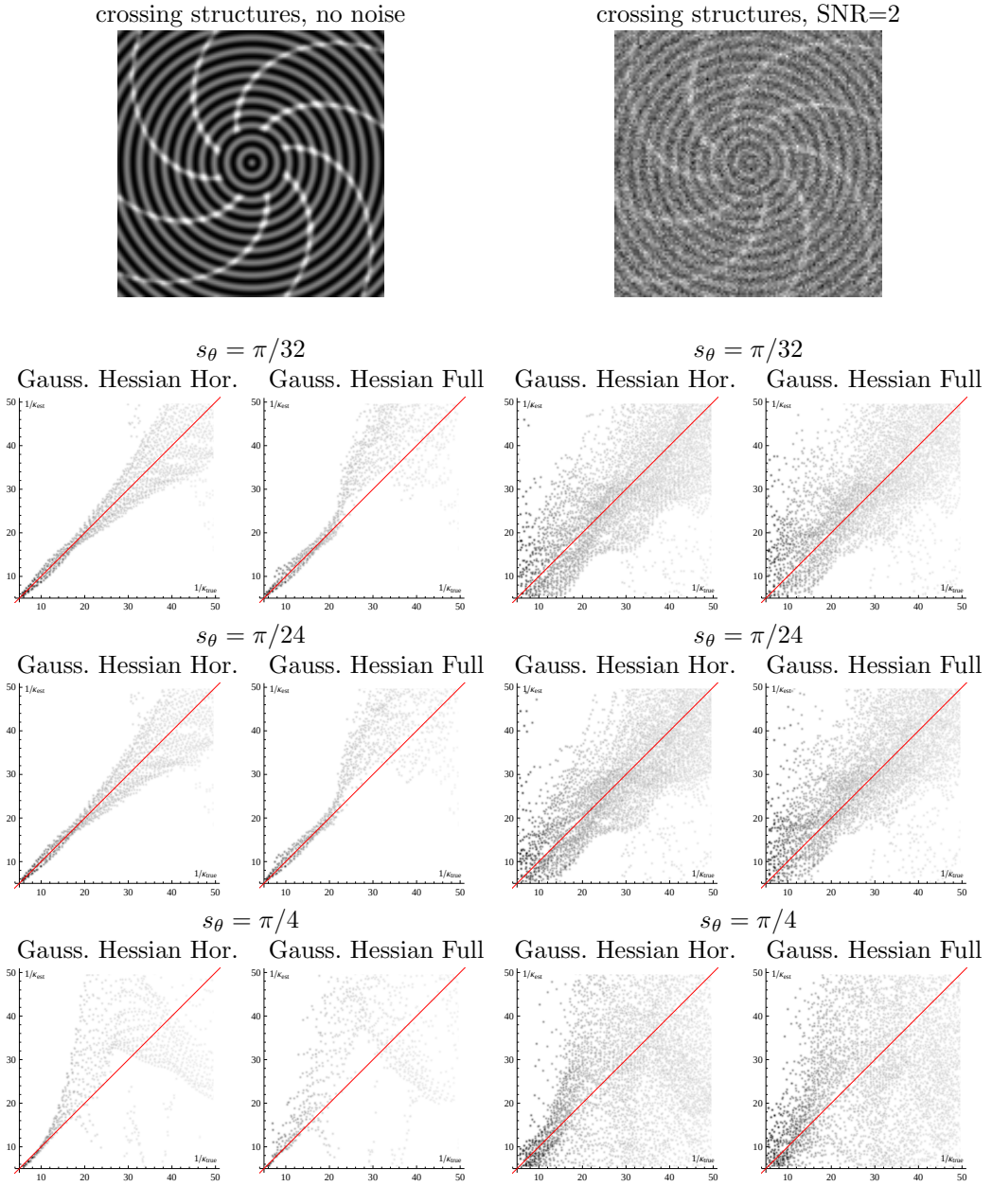


Figure 5.8: Curvature estimation on an image with crossing structures. See text (Subsection 5.5.4) for details.

from horizontality d_{H}, and orientation confidence s. Note, that these three scalar-valued features are a simple description for local curves; they do not completely describe the local second order derivative structure of the orientation score, which is captured by the 8 different scalars in the Hessian matrix cf. (5.10).

The experimental results show that the curvature estimates are in general reliable. However, it depends on the application which variant should be used. For images with lines and contours one can use the full Hessian method, which also provides an estimate for the deviation from horizontality. On the other hand, for images with oriented patterns it might be preferable to enforce horizontality.

If we knew what it was we were doing, it would not be called research, would it?

Albert Einstein (1879–1955)

6 Nonlinear Coherence-Enhancing Diffusion on Orientation Scores

This chapter is based on:

E. Franken, R. Duits, and B. ter Haar Romeny. Nonlinear diffusion on the 2D Euclidean motion group. In *Scale Space and Variational Methods in Computer Vision: Proceedings of the First International Conference, SSVM 2007, Ischia, Italy*, volume 4485 of LNCS, pages 461–472, Berlin, May–June 2007.

E.M. Franken and R. Duits. Crossing-preserving coherence-enhancing diffusion on invertible orientation scores. Accepted for publication in the *International Journal of Computer Vision (IJCV)*.

6.1 Introduction

Nonlinear diffusion is widely used in image processing. Perona and Malik [111] started this field of research by proposing an isotropic nonlinear diffusion technique where the diffusion is inhibited at edges. In this way, noise is reduced while edges remain sharp. Later, nonlinear *an*isotropic[1] diffusion equations have been studied, leading for instance to the well-known edge- and coherence-enhancing diffusion techniques [147]. The latter class of techniques improves the enhancement of edges resp. fibrous and line-like structures in images.

In Chapter 1 we already posed the problem of crossing and bifurcating structures. These cases are problematic for common nonlinear anisotropic diffusion techniques. Therefore, in this chapter we propose an anisotropic nonlinear diffusion method *on orientation scores*. The obvious advantage of our approach is that we can handle crossing structures appropriately, as shown in Figure 6.1.

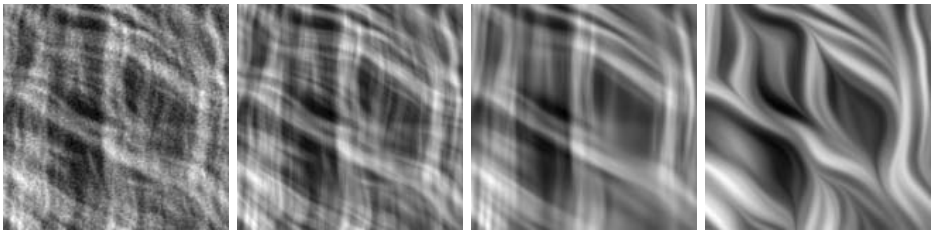
We will first pose the requirements that our diffusion process should fulfill. This leads to a definition of a nonlinear adaptive diffusion equation on $SE(2)$ that adapts the diffusion to the local curvature and deviation from horizontality (see Chapter 5) of elongated structures. Subsequently, two explicit numerical finite difference schemes will be presented to solve the diffusion equation. We will derive stability bounds for these schemes. Finally, we will show results of coherence-enhancing diffusion on orientation scores on artificial and biomedical images with crossing and curved elongated structures.

This chapter uses many results from previous chapters, especially from Chapters 2 and 5. For the sake of clarity, Figure 6.2 provides a schematic overview of the entire algorithm, including references to the appropriate sections in this thesis.

6.1.1 Related work

Many methods for enhancing elongated structures are based on nonlinear anisotropic diffusion equations on the image. This idea was pioneered by Nitzberg et al. [104] and Cottet et al. [26]. Later on, Weickert proposed edge- and coherence-enhancing diffusion filtering [146, 147], which uses the structure tensor to steer the diffusion process. The coherence-enhancing diffusion method solves the diffusion equation

[1] In literature, ambiguity arises for the term "anisotropic"; sometimes isotropic diffusion processes that are made adaptive to the local structure are called to be anisotropic. Here, we use the term anisotropic for diffusion where an unequal amount of diffusion is applied in different directions.



(a) Original (b) Our method $t = 2$ (c) Our method $t = 30$ (d) Standard CED $t = 30$

Figure 6.1: Example of coherence-enhancing diffusion on orientation scores. The image is constructed from two rotated 2-photon microscopy images of collagen tissue of a tissue-engineered heart valve. At $t = 2$ our method achieves a good enhancement of the image. Comparing the result at $t = 30$ with standard coherence-enhancing diffusion (CED), we can clearly see the improved performance for crossing structures.

on $\mathbb{R}^2$ (see eq. (2.80) on page 41), where the diffusion tensor is given by

$$\begin{aligned} \mathbf{D}(\boldsymbol{x}) &= \lambda_1 \, \boldsymbol{e}_1(\boldsymbol{x}) \, \boldsymbol{e}_1(\boldsymbol{x})^{\mathrm{T}} + \lambda_2(\boldsymbol{x}) \, \boldsymbol{e}_2(\boldsymbol{x}) \, \boldsymbol{e}_2(\boldsymbol{x})^{\mathrm{T}}, \quad \text{with } \lambda_1 = \alpha, \quad \text{and} \\ \lambda_2(\boldsymbol{x}) &= \begin{cases} \alpha & \text{if } \mu_1(\boldsymbol{x}) = \mu_2(\boldsymbol{x}); \\ \alpha + (1-\alpha) \exp\left(\frac{-C}{(\mu_1(\boldsymbol{x}) - \mu_2(\boldsymbol{x}))^2}\right) & \text{otherwise,} \end{cases} \end{aligned} \tag{6.1}$$

where $C > 0$ controls the nonlinear behavior, $0 < \alpha \ll 1$ is a small parameter that ensures that the diffusion tensors are positive definite, $\mu_1(\boldsymbol{x})$ and $\mu_2(\boldsymbol{x})$ are the eigenvalues of the structure tensor with $\mu_1(\boldsymbol{x}) > \mu_2(\boldsymbol{x})$, and $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are the corresponding eigenvectors. The structure tensor is given by

$$\mathcal{S}f(\boldsymbol{x}) = \left\{ G_\rho * \left(\begin{pmatrix} \partial_x (f * G_\sigma) \\ \partial_y (f * G_\sigma) \end{pmatrix} \cdot \begin{pmatrix} \partial_x (f * G_\sigma) \\ \partial_y (f * G_\sigma) \end{pmatrix}^{\mathrm{T}} \right) \right\} (\boldsymbol{x}). \tag{6.2}$$

Many publications have appeared inspired by coherence-enhancing diffusion and related methods, which all applied the nonlinear diffusion directly on the image. Besides by the work of Weickert, our approach has been inspired by Manniesing et al. [94, 95], who proposed to steer the diffusion using the vessel resemblance function, which is based on the Hessian instead of the structure tensor. Furthermore, Tschumperlé [134] proposed the idea to include curvature into the diffusion process in order to improve enhancement of curved structures.

An interesting alternative approach for handling crossing elongated structures is proposed by Scharr [121]. The method analyzes crossing oriented patterns using higher order derivatives using the approach of [130]. Instead of the gradient, the second order jet operator is used to both calculate a structure tensor and for the gradient and divergence of a diffusion-like PDE. The drawback of this method

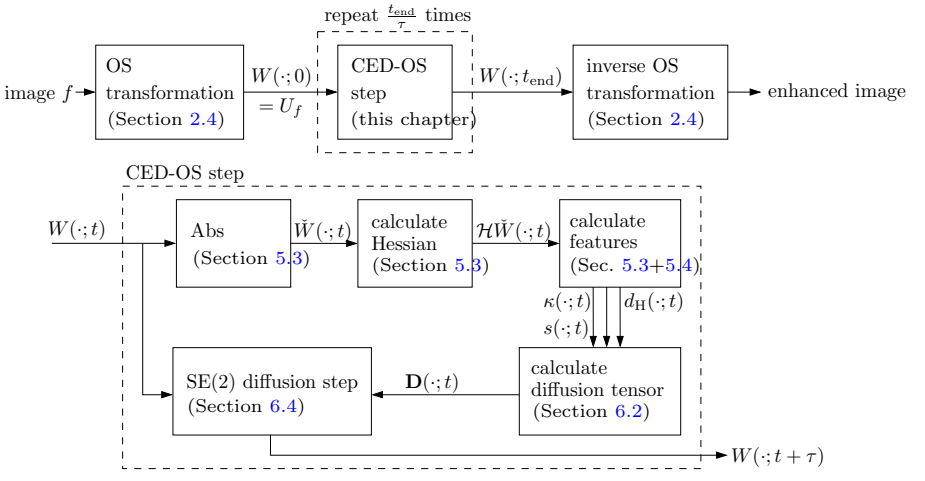


Figure 6.2: Flow chart of the CED-OS (Coherence-Enhancing Diffusion in Orientation Score) method.

is that the order of the PDE high, and gets even higher if one wants to deal with crossings of more than two elongated structures. Furthermore, it is only suitable to handle crossing curves with a large crossing angle. Our approach is more general, in the sense that the amount of crossing curves we can deal with is determined by the number of sampled orientations.

In wavelet literature, several authors have investigated image enhancement via wavelet transforms that are both *multi-orientation* and *multi-scale*. The *curvelet transform* [19, 18, 127] is a discrete multi-scale and multi-orientation wavelet transform. Antoine et al. propose continuous *directional wavelets* [4, 5] and establish the wavelet theory on the *similitude group*. The main difference with our approach is that the orientation score wavelet transform leaves out[2] scaling, while we are still able to reconstruct the image, as described in Section 2.4. Another difference is that instead of applying soft thresholds on the wavelet domain, which is common practice in wavelet literature, we explicitly employ the $SE(2)$ group structure in the wavelet domain. In fact, an orientation score wavelet picks up all image scales simultaneously[3]. The notion of scale comes into play when considering processing operations on orientation scores, as the diffusion time of the left-invariant diffusion equations on $SE(2)$ can be interpreted as a scale parameter.

[2]Besides the practical advantage of a reduction of storage requirements, another reason to leave out scaling is that the admissibility conditions for multi-scale multi-orientation wavelet transforms [5] require the oriented wavelet to oscillate along its radial direction. This is an undesirable effect that will cause problems for the diffusion processes that we aim at.

[3]I.e. all scales within a reasonable range determined by the sampling and the image size

6.2 Requirements on Coherence-Enhancing Diffusion via Orientation Scores

In this chapter we are interested in the nonlinear diffusion equation given by (cf. (2.81))

$$\begin{cases} \partial_t W(g,t) = \nabla \cdot (\mathbf{D}(W)(g,t)\, \nabla W(g,t)), \quad g \in SE(2), t \geq 0, \\ W(g,0) = U(g). \end{cases} \tag{6.3}$$

To come to a feasible way to construct the diffusion tensor, next we formulate a number of additional requirements.

Firstly, oriented structures should be enhanced while noise should be reduced. For this purpose, non-oriented regions should be smoothed isotropically, while oriented regions should be smoothed anisotropically, taking into account the local orientation and curvature. This can be accomplished by an adaptive mixture of diffusion along the exponential curve and μ-isotropic diffusion, cf. (2.87) on page 43

$$\mathbf{D}(W)(g) = (1 - D_{\mathrm{a}}) \frac{\mu^2}{\|\boldsymbol{c}\|_\mu^2} \boldsymbol{c}\, \boldsymbol{c}^{\mathrm{T}} + D_{\mathrm{a}}\, \mathrm{diag}(1, 1, \mu^2), \quad 0 < D_{\mathrm{a}} < 1, \tag{6.4}$$

where, for the current nonlinear adaptive case, we have to substitute $D_{\mathrm{a}} = D_{\mathrm{a}}(s(W(\cdot,t))$ and $\boldsymbol{c} = \boldsymbol{c}(W(\cdot,t))(g)$. The tangent vectors $\boldsymbol{c}$ can be estimated using the method described in Section 5.3 and $D_{\mathrm{a}}(s(W(\cdot,t))(g)) \in [0,1]$ should be inversely proportional to a measure for orientation confidence, viz. the one that is described in Section 5.4. Notice that if $D_{\mathrm{a}} = 1$, the dependency on $\boldsymbol{c}$ in (6.4) drops out, which is desirable since on non-oriented positions the local tangent vector, and thus the curvature and deviation from horizontality, is undefined.

Secondly, crossing structures should be preserved. This is automatically accomplished by using orientation scores. The smallest crossing angle that can be preserved depends on the number of orientations N_o being used, as well as on the parameters of the orientation score transformation.

Thirdly, we aim at enhancement of lines, contours, and oriented patterns (see Figure 1.6 on page 9) simultaneously, in order to make the algorithm described here generally applicable for many different types of images. Of course, one might want to tailor the algorithm later on to a specific application that only demands enhancement of one of these elongated structures. To ensure uniform handling of these elongated structures, the calculation of the diffusion tensor should be performed in a phase-invariant way, as described in Section 5.3. Furthermore, we require *average grey-value invariance*. In accordance to equation (5.6) in Section 5.3, we specify the initial condition of (6.3) as

$$W(g,0) = U_{\tilde{f}}(g), \quad \text{where } \tilde{f} = f - (f *_{\mathbb{R}^2} G_{\sigma_s}), \tag{6.5}$$

and the evolving feature orientation score that will be used to calculate the diffusion tensor is defined by

$$\check{W}(g,t) = |W(g,t)|\,, \quad \text{for all } g \in SE(2) \text{ and } t \in \mathbb{R}^+. \tag{6.6}$$

Finally, *grey-scaling invariance* is guaranteed to be preserved by applying a global normalization of the values for orientation confidence $s(W(\cdot,t))$, i.e. the calculation of D_{a} can be written as

$$D_{\mathrm{a}}(W(\cdot,t)) = \zeta\left(\frac{s(W(\cdot,t))}{\max\limits_{g\in SE(2)} s(W(g,t))}\right). \tag{6.7}$$

where $\zeta : \mathbb{R} \to [0,1]$ is a nonlinear function controlling the isotropy of the diffusion as function of the orientation confidence. The global normalization has the additional advantage that this function is easier to tune, as the range of values is independent on the scale parameters of the Hessian.

The nonlinear function ζ should have high orientation confidence if $D_{\mathrm{a}} \approx 0$ and small or negative orientation confidence gives $D_{\mathrm{a}} \approx 1$. Furthermore, the function should be continuous. In theory it should also be infinitely differentiable, but in practice this is not necessary since finite differences that will be used in the numerical schemes will anyway be bounded. Therefore, a large number of functions are possible, for instance

1. Exponential functions, given by

$$\zeta(s) = \begin{cases} 1 & s \leq 0; \\ 1 - \exp\left(-\left(\frac{c}{s}\right)^n\right) & n > 0 \text{ and } s > 0; \\ \exp\left(-\left(\frac{c}{s}\right)^n\right) & n < 0 \text{ and } s > 0, \end{cases}, \tag{6.8}$$

   with $n \in \mathbb{Z}\backslash\{0\}$, $c \in \mathbb{R}^+$. The case $n = -1$ coincides with the nonlinear function that we proposed in earlier work [54, 53]. The case $n = -2$ corresponds to one of the nonlinear functions that Perona and Malik [111] apply on the gradient magnitude. The cases $n > 0$ correspond to the nonlinear functions that Weickert [146] proposes for Perona and Malik-like diffusion (usually with $n = 4$), coherence-enhancing diffusion (default $n = 2$) and edge-enhancing diffusions (default $n = 4$). The function is infinitely differentiable everywhere for the cases $n > 0$. For the cases $n < 0$, at $s = 0$ the function is only $-n - 1$ times differentiable.

2. Fractional functions, given by

$$\zeta(s) = \begin{cases} \frac{1}{\left(q+\left(\frac{s}{c}\right)^n\right)^p} & s \geq 0; \\ 1 & \text{otherwise,} \end{cases} \qquad n \in \mathbb{N},\ p \in \mathbb{R}^+,\ c \in \mathbb{R}^+,\ q \in \mathbb{R}^+. \tag{6.9}$$

The case $q = 1$, $n = 2$, and $p = 1$ coincides with the other type of diffusivities proposed by Perona and Malik [111]. The case $q = 1$, $n = 2$, and $p = 1/2$ coincides with the diffusivity of Charbonnier et al. [20]. The case $q \to 0$, $n = 1$, and $p = 1$ coincides with total variation flow [15]. This class of functions is infinitely differentiable everywhere except at $s = 0$ where it is $n - 1$ times differentiable.

The different classes of functions have different theoretical properties. Most importantly, the exponential function for $n > 0$ tend to sharpen edges since backward diffusion can occur, while the fractional functions do not possess this property [145]. From the practical point of view, however, on orientation scores there does not seem to be a strong reason to prefer one function over the other. Which one performs best is application dependent. In the experimental section (Section 6.5), we will therefore choose one of the functions purely based on experimental results.

6.3 Gauge Frame Interpretation of Anisotropic Diffusion in $SE(2)$

The idea of gauge coordinates in image processing [47] is to define a data-dependent orthogonal coordinate frame (the gauge frame) for the tangent space $T_{\boldsymbol{x}}(\mathbb{R}^n)$ at each position $\boldsymbol{x}$, such that the basis vectors are in alignment with some local feature of interest in the image. The advantage of gauge coordinates is that operations described in gauge coordinates are automatically rotation invariant. The most commonly used gauge coordinates are determined by the gradient of a 2D image, where one basis vector is fixed tangent and one is fixed orthogonal to the direction of the gradient.

The introduced nonlinear diffusion can be seen as a diffusion process in which the eigenvectors of the diffusion process are aligned with a second order gauge frame. In fact, the method establishes a μ-*orthogonal* gauge frame $\{\partial_a, \partial_b, \partial_c\}$, where μ-orthogonality is defined as

$$(\partial_p, \partial_q)_\mu = \mu^2 \delta_{pq}, \quad p, q \in \{a, b, c\}. \tag{6.10}$$

Since exponential curves provide a local description of the structures we are interested in, i.e. curved elongated structures, one of the components of the gauge frame should be established by the tangent vector $\boldsymbol{c}(g)$ at each position $g \in SE(2)$ with $\|\boldsymbol{c}(g)\|_\mu = 1$, i.e. (omitting the dependency on g)

$$\partial_a = \mu\, c^\theta \partial_\theta + c^\xi \partial_\xi + c^\eta \partial_\eta. \tag{6.11}$$

The other two components ∂_b and ∂_c should span the plane orthogonal to ∂_a, and therefore cannot be uniquely defined. We make the an arbitrary, but unique

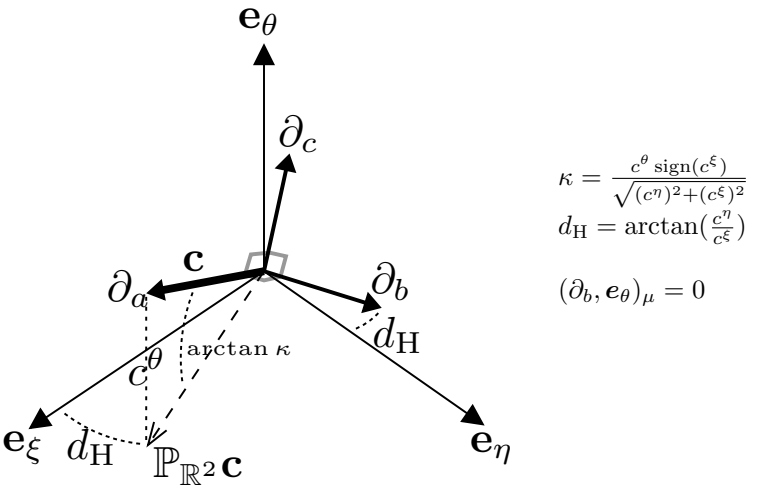


Figure 6.3: Illustration of curvature and deviation from horizontality (Subsection 2.8.7) and of the gauge frame (Section 6.3). Note that for visualization reasons, the lengths of the vectors are arbitrary. The true lengths are given by $\|\boldsymbol{c}\|_\mu = \|\boldsymbol{e}_\theta\|_\mu = \|\boldsymbol{e}_\xi\|_\mu = \|\boldsymbol{e}_\eta\|_\mu = 1$ and $\|\partial_a\|_\mu = \|\partial_b\|_\mu = \|\partial_c\|_\mu = \mu$.

choice by ensuring that $\{\partial_a, \partial_b, \partial_c\}$ coincides with the left-invariant coordinate frame $\{\partial_\xi, \partial_\eta, \partial_\theta\}$ for the case of straight horizontal lines ($\kappa = 0$ and $d_{\mathrm{H}} = 0$), i.e.

$$\partial_a = \partial_\xi, \quad \partial_b = \partial_\eta, \quad \partial_c = \mu\partial_\theta, \quad \text{iff } \boldsymbol{c} = (0,1,0). \tag{6.12}$$

In terms of κ and d_{H} (as defined in Subsection 2.8.7 on page 39) this renders the following gauge frame

$$\begin{pmatrix} \partial_a \\ \partial_b \\ \partial_c \end{pmatrix} = \mathbf{Q}_{\kappa,\mu}^{\mathrm{T}} \tilde{\mathbf{R}}_{d_{\mathrm{H}}}^{\mathrm{T}} \begin{pmatrix} \partial_\xi \\ \partial_\eta \\ \mu\partial_\theta \end{pmatrix}, \tag{6.13}$$

where

$$\tilde{\mathbf{R}}_{d_{\mathrm{H}}} = \begin{pmatrix} \cos d_{\mathrm{H}} & -\sin d_{\mathrm{H}} & 0 \\ \sin d_{\mathrm{H}} & \cos d_{\mathrm{H}} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \mathbf{Q}_{\kappa,\mu} = \begin{pmatrix} \frac{\mu}{\sqrt{\mu^2+\kappa^2}} & 0 & \frac{\kappa}{\sqrt{\mu^2+\kappa^2}} \\ 0 & 1 & 0 \\ \frac{-\kappa}{\sqrt{\mu^2+\kappa^2}} & 0 & \frac{\mu}{\sqrt{\mu^2+\kappa^2}} \end{pmatrix}. \tag{6.14}$$

This gauge frame is illustrated in Figure 6.3. Note that it actually involves two rotations, since $\mathbf{Q}_{\kappa,\mu}$ is also a rotation matrix with rotation angle $\arccos \frac{\mu}{\sqrt{\mu^2+\kappa^2}}$.

The class of nonlinear $SE(2)$ diffusions cf. (6.4) can now be expressed in the gauge

coordinates as

$$\begin{cases} \partial_t W(g,t) = \Big(\partial_a \partial_a + \partial_b(D_{\mathrm{a}}\partial_b) + \partial_c(D_{\mathrm{a}}\partial_c)\Big) W(g,t), \\ W(g,t=0) = U_f(g). \end{cases} \tag{6.15}$$

In left-invariant derivatives this equation can now be written as

$$\begin{aligned} &\partial_t W(g,t) \\ &\quad = \begin{pmatrix} \partial_\xi & \partial_\eta & \mu\partial_\theta \end{pmatrix} \left\{ \tilde{\mathbf{R}}_{d_{\mathrm{H}}} \mathbf{Q}_{\kappa,\mu} \begin{pmatrix} 1 & 0 & 0 \\ 0 & D_{\mathrm{a}} & 0 \\ 0 & 0 & D_{\mathrm{a}} \end{pmatrix} \mathbf{Q}_{\kappa,\mu}^{\mathrm{T}} \tilde{\mathbf{R}}_{d_{\mathrm{H}}}^{\mathrm{T}} \right\} \begin{pmatrix} \partial_\xi \\ \partial_\eta \\ \mu\partial_\theta \end{pmatrix} W(g,t). \end{aligned} \tag{6.16}$$

This equation coincides with the diffusion process cf. (6.4).

6.4 Numerical Schemes for Nonlinear Diffusion

In this section we will propose two explicit finite difference schemes to solve diffusion equation (6.3). We restrict to explicit schemes since implicit schemes are in general computationally expensive for our 3-dimensional anisotropic case. Furthermore, our anisotropic diffusion process requires good rotational invariance. Many efficient (semi)-implicit schemes with operator splitting, e.g. the additive operator splitting (AOS) scheme [147], are therefore discarded due to their poor rotation invariance (see Figure 6.4). The LSAS scheme [150] has good rotational invariance and can be performed in 3D, however it is inherently designed for isotropic grids, which is problematic in our case since we do not have a natural notion of isotropy. One could use the artificial notion of μ-isotropy and use LSAS, but this is only possible if we restrict to orientation score "voxels" that are μ-isotropic, and there is no good reason for this restriction. Orientation score voxels are defined to be μ-isotropic if the distance, with regard to the μ-norm of Subsection 2.8.4, between two spatially neighboring sampling points is equal to the distance between two orientationally neighboring sampling points. This is only the case if $\mu = s_\theta$, where s_θ denotes the sample distance in the orientation dimension in radians.

6.4.1 Simple Explicit Finite Difference Scheme

A simple scheme for solving the nonlinear diffusion PDE (6.3) is obtained by rewriting the PDE to ∂_x, ∂_y, and ∂_θ derivatives, and using centered finite differences to calculate the first order derivatives, e.g. for second order accurate finite

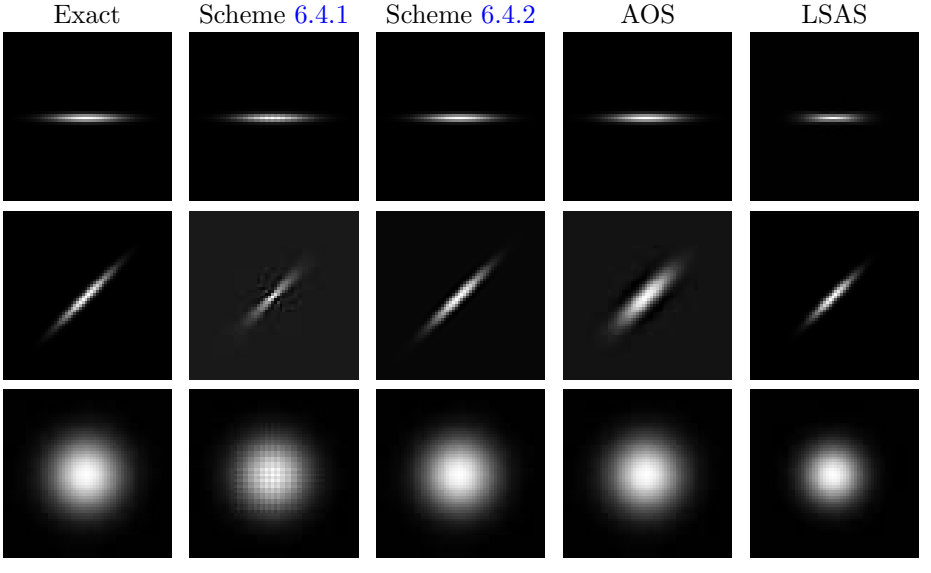


Figure 6.4: Comparison of rotation invariance of different numerical anisotropic diffusion schemes. The input to all algorithms is an image with dimensions 60×60 with a single Gaussian blob with scale 0.9 pixels, linear diffusion is applied with diffusion tensor $\mathbf{D} = \mathbf{R}_\theta \operatorname{diag}(D_{xx}, D_{yy}) \mathbf{R}_\theta^{-1}$, $\tau = 0.25$, and end time $t = 35$. The exact solution is found simply by convolving the input image with the anisotropic Gaussian kernel. Top row: $\theta = 0$, $D_{xx} = 1$, $D_{yy} = 0.0025$. Middle row: $\theta = \pi/4$, $D_{xx} = 1$, $D_{yy} = 0$. Bottom row: $D_{xx} = D_{yy} = 1$ (isotropic diffusion).

differences this yields

$$\begin{aligned} \partial_x W^k(\boldsymbol{x}, l) &\approx \frac{1}{2} \left(W^k(\boldsymbol{x} + \boldsymbol{e}_x, l) - W^k(\boldsymbol{x} - \boldsymbol{e}_x, l) \right), \\ \partial_y W^k(\boldsymbol{x}, l) &\approx \frac{1}{2} \left(W^k(\boldsymbol{x} + \boldsymbol{e}_y, l) - W^k(\boldsymbol{x} - \boldsymbol{e}_y, l) \right), \\ \partial_\theta W^k(\boldsymbol{x}, l) &\approx \frac{1}{2 s_\theta} \left(W^k(\boldsymbol{x}, l+1) - W^k(\boldsymbol{x}, l-1) \right). \end{aligned} \tag{6.17}$$

where $k \in \mathbb{N}$ denotes the discrete time and $(l \mod N_o) \in [0, N_o - 1]$ denotes the sampled orientation axis where $\theta = l\, s_\theta$ with orientation sample distance s_θ. In the time direction we use the first order forward finite difference $(W^{k+1} - W^k)/\tau$ where τ the time step. The advantage of this scheme is the efficiency and good stability since the effective stencil size is 5 in each dimension rather than 3 in most other simple explicit schemes. The drawback, however, is that oscillations at the Nyquist frequency can occur, caused by the fact that a concatentation of two first order centered differences gives $\partial_x^2 W = \frac{1}{4} \left(W(\boldsymbol{x} + 2\boldsymbol{e}_x, l) - 2W(\boldsymbol{x}, l) + W(\boldsymbol{x} - 2\boldsymbol{e}_x, l) \right)$, i.e. the closest neighboring pixels are not taken into account. The latter problem can be resolved by adding some additional coupling between neighboring pixels,

for instance by using a 3-pixel scheme to perform the isotropic part of the diffusion [144].

6.4.1.1 Stability Analysis

For stability analysis, we consider the linear diffusion equation (6.3) assuming a constant anisotropy factor D_a with $0 \leq D_\mathrm{a} \leq 1$ and find an upper bound for time step τ such that the equation remains stable for all applicable cases. In $\{\partial_\theta, \partial_x, \partial_y\}$-coordinates, the diffusion tensor components are given by

$$\begin{aligned}
D_{xx} &= \cos^2(\alpha)\cos^2(\theta) + D_\mathrm{a}\left(\cos^2(\theta)\sin^2(\alpha) + \sin^2(\theta)\right),\\
D_{yy} &= D_\mathrm{a}\cos^2(\theta) + \left(\cos^2(\alpha) + D_\mathrm{a}\sin^2(\alpha)\right)\sin^2(\theta),\\
D_{\theta x} &= \frac{1}{q}(D_\mathrm{a} - 1)\cos(\alpha)\cos(\theta)\sin(\alpha),\\
D_{\theta y} &= \frac{1}{q}(D_\mathrm{a} - 1)\cos(\alpha)\sin(\alpha)\sin(\theta),\\
D_{xy} &= (1 - D_\mathrm{a})\cos^2(\alpha)\cos(\theta)\sin(\theta),\\
D_{\theta\theta} &= \frac{1}{q^2}\left(D_\mathrm{a}\cos^2(\alpha) + \sin^2(\alpha)\right),
\end{aligned} \tag{6.18}$$

where $q = \frac{s_\theta}{\mu}$, $\sin\alpha = \frac{\kappa}{\sqrt{\mu^2+\kappa^2}}$, and $\cos\alpha = \frac{\mu}{\sqrt{\mu^2+\kappa^2}}$. The first order finite differences are defined in (6.17), rendering the following stencils for the second order finite differences that are applied in all three dimensions

$$\mathbf{S}_{ii} = \frac{1}{4}\begin{pmatrix}1 & 0 & -2 & 0 & 1\end{pmatrix}, \qquad \mathbf{S}_{ij|i\neq j} = \frac{1}{4}\begin{pmatrix}-1 & 0 & 1\\ 0 & 0 & 0\\ 1 & 0 & -1\end{pmatrix}. \tag{6.19}$$

One numerical iteration can be written as

$$W^{k+1} = W^k + \tau\mathbf{A}(W^k)W^k = \underbrace{(\mathbf{I} + \tau\mathbf{A}(W^k))}_{\mathbf{M}(W^k)}W^k, \tag{6.20}$$

where $\mathbf{M}(W^k)$ is a square matrix with size equal to the total number of "voxels" in the orientation score. The numerical method is stable as long as the absolute values of all eigenvalues of $\mathbf{M}$ are ≤ 1.

The Gershgorin circle theorem [58] states that for a complex- or real-valued $n \times n$ matrix $\mathbf{M}$, each eigenvalue is in at least one of the disks given by $\{z : z \in \mathbb{C}, |z - C_i| \leq R_i\}$ where the center $C_i = M_{ii}$ and the radius $C_i = \sum_{j=1, i\neq j}^{n} |M_{ij}|$. Using this theorem we find that all eigenvalues are situated in a circle with center C and radius R given by

$$\begin{aligned}
C &= 1 - \frac{\tau}{2}(D_{xx} + D_{yy} + D_{\theta\theta})\\
R &= \frac{\tau}{2}(|D_{xx}| + |D_{yy}| + 4|D_{\theta x}| + 4|D_{\theta y}| + 4|D_{xy}| + |D_{\theta\theta}|).
\end{aligned} \tag{6.21}$$

Stability requires $-1 \leq C - R$ and $C + R \leq 1$. The first inequality gives as bound for τ

$$\begin{aligned}\tau \leq 4q^2 \Big\{ & (1+q^2) + D_{\mathrm{a}}(1+3q^2) + \cos(2\alpha)(D_{\mathrm{a}}-1)(1-q^2) \\ & + 2q\,|(D_{\mathrm{a}}-1)\sin(2\alpha)|\;(|\cos\theta| + |\sin\theta|) \\ & + q^2\,|(1-D_{\mathrm{a}})\sin(2\theta)(\cos(2\alpha)+1)| \Big\}^{-1}.\end{aligned} \tag{6.22}$$

Note that the second inequality $C + R \leq 1$ never holds as (6.21) shows that $C + R \geq 1$. However it can safely be discarded since $\mathbf{A}(W^k)$ is negative definite i.e. $\sigma(\mathbf{A}(W^k)) < 0$, where σ denotes the spectrum of the matrix, implying $\sigma(\mathbf{M}(W^k)) = \sigma(\mathbf{I} + \tau\mathbf{A}(W^k)) = 1 + \tau\sigma(\mathbf{A}(W^k)) < 1$ for all $\tau > 0$.

We want to find the case for which we get the lowest upper bound for τ, to guarantee stability in all circumstances. For the sine and cosine terms in the denominator, these worst case values are

$$\begin{aligned}\cos(2\alpha) = 1, \quad & \text{for } \alpha = \frac{\pi}{4}, \\ \sin(2\alpha) = 1, \quad & \text{for } \alpha = 0, \\ (|\cos\theta| + |\sin\theta|) = \sqrt{2}, \quad & \text{for } \theta = \frac{\pi}{4}.\end{aligned} \tag{6.23}$$

Furthermore, for D_{a} we have to set $D_{\mathrm{a}} = 0$ to get the "worst case" for τ, since degenerate anisotropic diffusion is most critical concerning stability. So, we find the following upper bound for τ, which guarantees stability for all θ, α, and $0 \leq D_{\mathrm{a}} \leq 1$

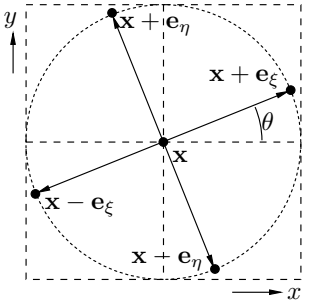
$$\tau \leq \frac{4q^2}{1 + 2\sqrt{2}q + 3q^2 - |1-q^2|}. \tag{6.24}$$

For a typical value in practice of $q = \frac{s_\theta}{\mu} = \frac{\pi/32}{0.1}$ this yields $\tau \leq 0.60$, which is a fairly sharp upper bound if we consider our practical observations.

6.4.2 Left-Invariant Explicit Scheme with Spline Interpolation

The important property of the differential operators ∂_ξ, ∂_η, and ∂_θ is their left-invariance. The performance of a numerical scheme might therefore be more optimal if this left-invariance is carried over to the finite differences that are used. To achieve this we should define the spatial finite differences in the directions defined by the left-invariant basis vectors $\{\boldsymbol{e}_\xi, \boldsymbol{e}_\eta\}$, instead of the basis vectors $\{\boldsymbol{e}_x, \boldsymbol{e}_y\}$ that are aligned to the sample grid. In effect, the principal axes of diffusion in the spatial plane are always aligned with the finite differences as long as we only consider the case $d_{\mathrm{H}} = 0$. The reason that we do not consider

nonzero deviation from horizontality is that this scheme becomes expensive and complicated in that case.



$$
\begin{aligned}
\partial_\xi W &\approx \frac{1}{2}\Big(W(\boldsymbol{x}+\boldsymbol{e}_\xi^l, l) - W(\boldsymbol{x}-\boldsymbol{e}_\xi^l, l)\Big)\\
\partial_\xi^2 W &\approx W(\boldsymbol{x}+\boldsymbol{e}_\xi^l, l) - 2W(\boldsymbol{x}, l) + W(\boldsymbol{x}-\boldsymbol{e}_\xi^l, l)\\
\partial_\eta W &\approx \frac{1}{2}\Big(W(\boldsymbol{x}+\boldsymbol{e}_\eta^l, l) - W(\boldsymbol{x}-\boldsymbol{e}_\eta^l, l)\Big)\\
\partial_\eta^2 W &\approx W(\boldsymbol{x}+\boldsymbol{e}_\eta^l, l) - 2W(\boldsymbol{x}, l) + W(\boldsymbol{x}-\boldsymbol{e}_\eta^l, l)\\
\partial_\theta W &\approx \frac{1}{2s_\theta}\left(W(\boldsymbol{x}, l+1) - W(\boldsymbol{x}, l-1)\right)\\
\partial_\theta^2 W &\approx \frac{1}{{s_\theta}^2}\left(W(\boldsymbol{x}, l+1) - 2W(\boldsymbol{x}, l) + W(\boldsymbol{x}, l-1)\right)
\end{aligned}
$$

$$
\partial_\xi \partial_\theta W \approx \frac{1}{4s_\theta}\left(W(\boldsymbol{x}+\boldsymbol{e}_\xi^l, l+1) - W(\boldsymbol{x}+\boldsymbol{e}_\xi^l, l-1) - W(\boldsymbol{x}-\boldsymbol{e}_\xi^l, l+1) + W(\boldsymbol{x}-\boldsymbol{e}_\xi^l, l-1)\right)
$$

$$
\partial_\theta \partial_\xi W \approx \frac{1}{4s_\theta}\left(W(\boldsymbol{x}+\boldsymbol{e}_\xi^{l+1}, l+1) - W(\boldsymbol{x}+\boldsymbol{e}_\xi^{l+1}, l-1) - W(\boldsymbol{x}-\boldsymbol{e}_\xi^{l-1}, l+1) + W(\boldsymbol{x}-\boldsymbol{e}_\xi^{l-1}, l-1)\right)
$$

Figure 6.5: Illustration of the spatial part of the stencil of the numerical scheme. The horizontal and vertical dashed lines indicate the sampling grid, which is aligned with $\{\boldsymbol{e}_x, \boldsymbol{e}_y\}$. The stencil points, indicated by the black dots, are aligned with the rotated left-invariant coordinate system $\{\boldsymbol{e}_\xi, \boldsymbol{e}_\eta\}$ with $\theta = l \cdot s_\theta$.

For the numerical scheme we apply the chain rule[4] on the right-hand side of the PDE (6.3) expressed in left-invariant derivatives cf. (6.16) with $d_{\mathrm{H}} = 0$. The left-invariant derivatives are replaced by the finite differences defined in Figure 6.5. In time direction we use the first order forward finite difference. Interpolation is required at spatial positions $\boldsymbol{x} \pm \boldsymbol{e}_\xi$ and $\boldsymbol{x} \pm \boldsymbol{e}_\eta$. For this purpose we use the algorithms for B-spline interpolation proposed by Unser et al. [136] with B-spline order 2. This interpolation algorithm consists of a prefiltering step with a separable infinite impulse response filter to determine the B-spline coefficients. The interpolation images such as $W^k(\boldsymbol{x} \pm \boldsymbol{e}_\xi)$ can then be calculated by a separable convolution with a shifted B-spline.

The main advantage of this scheme is the improved rotation invariance of, see Figure 6.4. The drawback, however, is the computational speed and the additional blurring caused by the interpolation scheme.

6.4.2.1 Stability Analysis

We derive a stability bound for the numerical scheme described above in the same fashion as for the numerical scheme described in Subsection 6.4.1. Again we

[4] I.e., analogue to 1D: $\partial_x(D\,\partial_x W) = D\,\partial_x^2 W + (\partial_x D)(\partial_x W)$.

assume $0 \leq D_{\mathrm{a}} \leq 1$. In $\{\partial_\theta, \partial_\xi, \partial_\eta\}$ coordinates, the diffusion tensor is given by

$$\begin{aligned} D_{\xi\xi} &= \cos^2(\alpha) + D_{\mathrm{a}} \sin^2(\alpha), \\ D_{\eta\eta} &= D_{\mathrm{a}}, \\ D_{\theta\xi} &= \frac{1}{q}(D_{\mathrm{a}} - 1)\cos(\alpha)\sin(\alpha), \\ D_{\theta\eta} &= D_{\xi\eta} = 0, \\ D_{\theta\theta} &= \frac{1}{q^2} D_{\mathrm{a}} \cos^2(\alpha) + \sin^2(\alpha), \end{aligned} \tag{6.25}$$

where $q = \frac{s_\theta}{\mu}$. The used stencils for the second order finite differences are

$$\mathbf{S}_{ii} = \begin{pmatrix} 1 & -2 & 1 \end{pmatrix}, \qquad \mathbf{S}_{ij|i\neq j} = \frac{1}{4}\begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}. \tag{6.26}$$

In this scheme, all off-center stencil positions are obtained by second order spline interpolation, which has to be taken into account in the stability analysis. In the matrix $\mathbf{M}$, with $W^{k+1} = \mathbf{M}(W^k)W^k$, this interpolation leads to more nonzero off-diagonal terms since interpolation amounts to a linear combination of a number of voxel values. Second order B-spline interpolation does not preserve the global maximum $m = \max_{g\in SE(2)}(W(g))$, but we can find a factor $1 < \chi < \infty$ which gives an upper bound such that $W_{\text{interpolated}}(g) \leq \chi \cdot m$ for all g i.e. for an arbitrary interpolation. This factor χ is found as follows

$$\chi = \max_{\Delta\in[-1/2,+1/2]} \sum_{x=-\infty}^{+\infty} |C_2(x+\Delta)| = \sqrt{2}, \tag{6.27}$$

where C_2 is the cardinal spline [136] corresponding to the second order B-spline, which is in fact the net convolution kernel that is used for spline interpolation.

Using the Gershgorin circle theorem and taking into account $\chi = \sqrt{2}$ we find for the circle center C and circle radius R

$$\begin{aligned} C &= 1 - 2\tau\left(\left(\frac{D_{\mathrm{a}}}{q^2} + 1\right)\cos^2(\alpha) + \left(D_{\mathrm{a}} + \frac{1}{q^2}\right)\sin^2(\alpha) + D_{\mathrm{a}}\right), \\ R &= 2\tau\Bigg(\sqrt{2}D_{\mathrm{a}} + \sqrt{2}\left|\frac{(D_{\mathrm{a}} - 1)\cos(\alpha)\sin(\alpha)}{q}\right| + \\ &\qquad \left(D_{\mathrm{a}} q^{-2} + \sqrt{2}\right)\cos^2(\alpha) + \left(q^{-2} + \sqrt{2}D_{\mathrm{a}}\right)\sin^2(\alpha)\Bigg), \end{aligned} \tag{6.28}$$

stability requires $-1 \leq C - R$ and $C + R \leq 1$. The first equation renders as bound for τ

$$\begin{aligned} \tau \leq \Bigg(&\frac{1}{\sqrt{2}}\left|\frac{(1 - D_{\mathrm{a}})\sin(2\alpha)}{q}\right| + q^{-2}\left(D_{\mathrm{a}} + (D_{\mathrm{a}} - 1)\cos(2\alpha) + 1\right) \\ &+ \frac{1}{2}\left(1 + \sqrt{2}\right)\left(3D_{\mathrm{a}} + (1 - D_{\mathrm{a}})\cos(2\alpha) + 1\right)\Bigg)^{-1}. \end{aligned} \tag{6.29}$$

By inserting the "worst case values" for all sine and cosine terms, see (6.23), and by setting isotropic diffusion $D_{\mathrm{a}} = 1$, which in this case is the worst-case for stability since it gives the largest off-diagonal components in matrix $\mathbf{M}$, we find the following upper bound for τ, which guarantees stability for all θ and α and $0 \leq D_{\mathrm{a}} \leq 1$

$$\tau \leq \frac{2q^2}{4 + 4(1+\sqrt{2})q^2}. \tag{6.30}$$

For a typical value in practice of $q = \frac{s_\theta}{\mu} = \frac{\pi/32}{0.1}$ this yields $\tau \leq 0.15$, which agrees with practical observations.

6.5 Results

In this section we compare the results of coherence-enhancing diffusion in the orientation score (CED-OS) cf. (6.3) with results obtained by the normal coherence-enhancing diffusion (CED) approach [147] where we use the LSAS numerical scheme [150].

The following parameters are used for the orientation score transformation (see Subsection 2.4.1 on page 19): $k = 2$, $q = 8$, $t = 1.6\left(\frac{X}{2\pi}\right)^2$ where X is the number of pixels of the image in x-direction, and $\sigma_s = 200$. These parameters are chosen such that the reconstruction is visually indistinguishable from the original. All orientation scores have a periodicity of π. The original images all have a range of

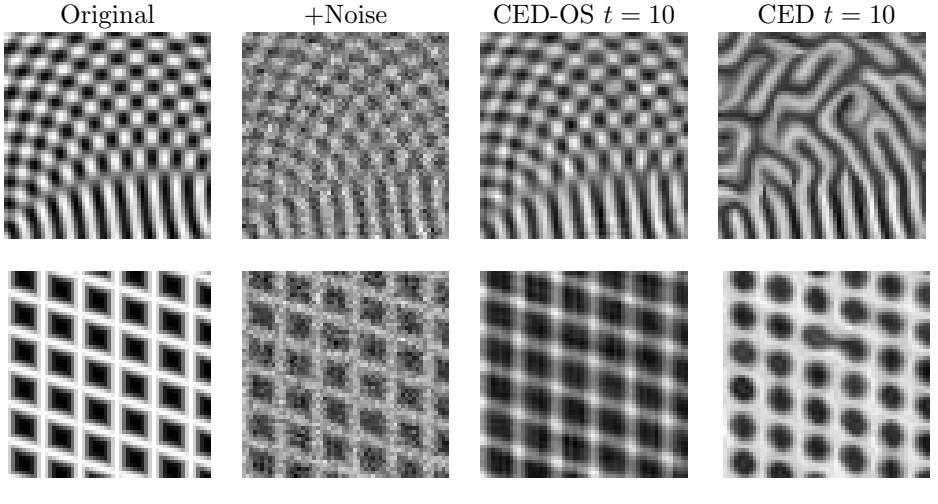
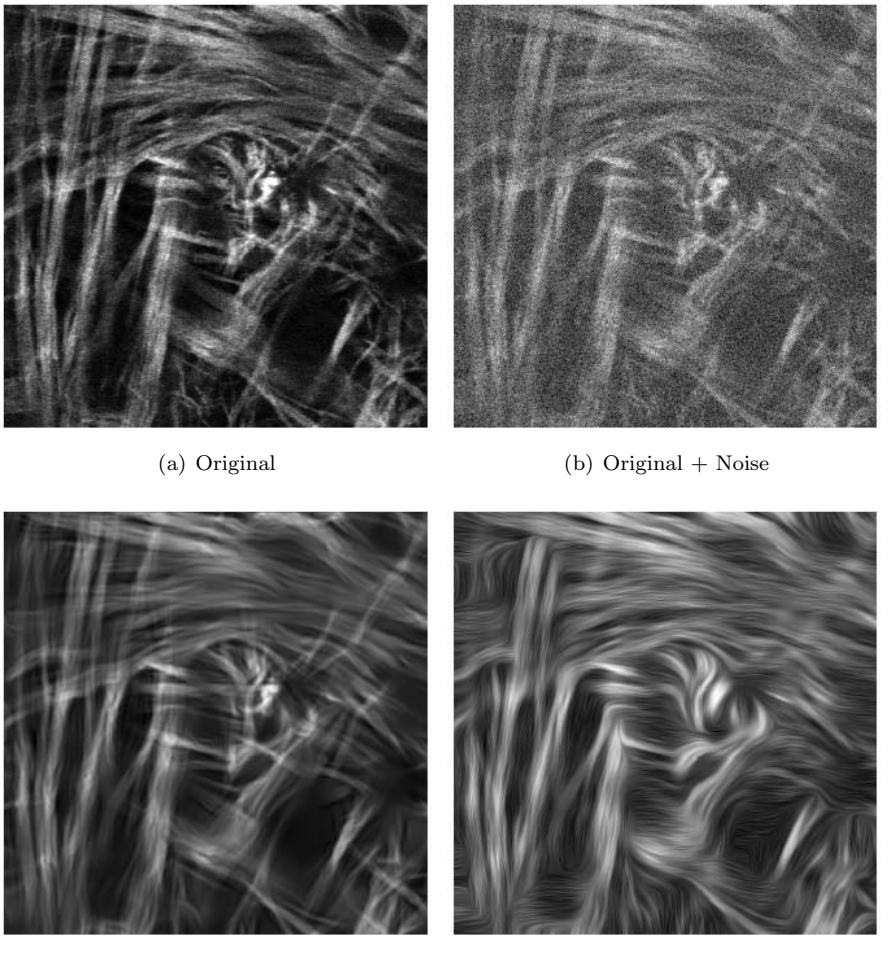


Figure 6.6: Shows the typical different behavior of CED-OS compared to CED. In CED-OS crossing structures are better preserved.



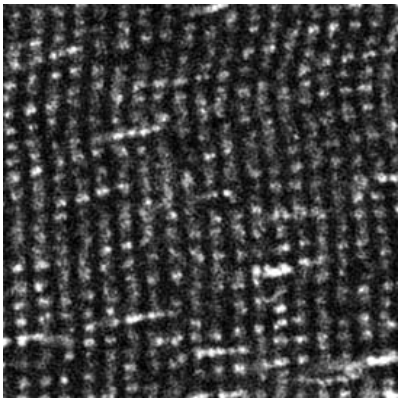
(a) Original (b) Original + Noise

(c) CED-OS $t = 30$ (d) CED $t = 30$

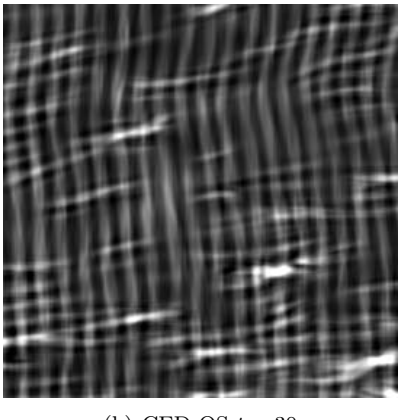
Figure 6.7: Result of CED-OS and CED on microscopy images of bone tissue. Additional Gaussian noise is added to evaluate the behavior on noisy images. Both the CED-OS and CED algorithm are applied on the noisy image. See Section 6.5 for details.

pixel values between 0 and 1. To ensure numerical stability, in the experiments where we use the numerical scheme of Subsection 6.4.1 we set $\tau = 0.25$ and for the numerical scheme of Subsection 6.4.2 we set $\tau = 0.1$. Note that the resulting images we will show of CED-OS do not represent the evolving orientation score, but only the reconstructed image after summation over all orientations.

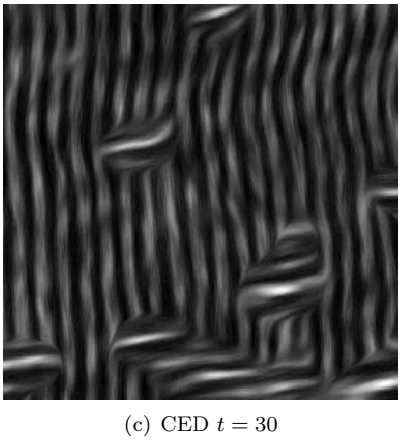
Figure 6.6 shows the effect of CED-OS compared to CED on artificial images with crossing line structures. The upper image shows an additive superposition



(a) Original



(b) CED-OS $t = 30$



(c) CED $t = 30$

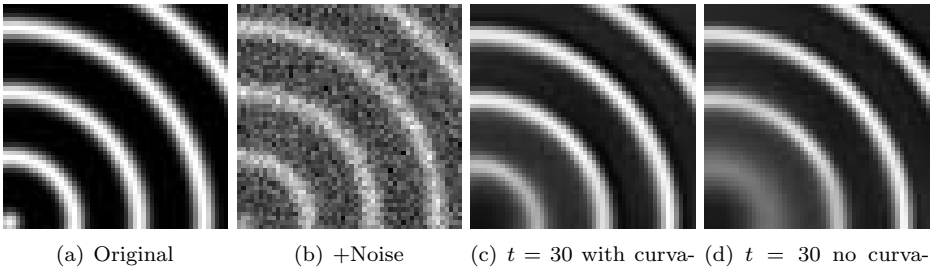
Figure 6.8: Result of CED-OS and CED on a microscopy image of a muscle cell. See Section 6.5 for details.

of two images with concentric circles. Our method is able to preserve the crossing structures, while CED cannot. The same holds for the lower image with crossing straight lines, where it should be noted that our method leads to amplification of the crossings, which is caused by the fact that the lines in the original image are not superimposed linearly. In this experiment, the diffusion process is not adapted using an estimate for deviation from horizontality (i.e. $d_{\mathrm{H}} = 0$), and the numerical scheme of Subsection 6.4.2 is used. The nonlinear diffusion parameters for CED-OS are: $N_o = 32$, $t_s = 12$, $\rho_s = 0$, $\mu = 0.058$, $c = 0.08$, and we use function (6.8) with $n = -1$. The parameters that we used for CED are (see [147]): $\sigma = 1$, $\rho = 1$, $C = 1$, and $\alpha = 0.001$. The images have a size of 56×56 pixels.

Figure 6.1 on page 135 shows the results on an image of collagen fibres obtained using 2-photon microscopy. This kind of images is acquired in tissue engineering research, where the goal is to create artificial heart valves [119]. All parameters during these experiments were set the same as the artificial images mentioned above except for CED parameter $\rho = 6$. The image size is 160×160 pixels.

Figures 6.7 and 6.8 show examples of the method on other microscopy data, see [49] resp. [124] for more information on the biomedical relevance of these images. The same parameters are used as above except for $t_s = 25$ in Figure 6.8. Clearly, the curve enhancement and noise suppression of the crossing curves is good in our method, while standard coherence-enhancing diffusion tends to destroy crossings and create artificial oriented structures.

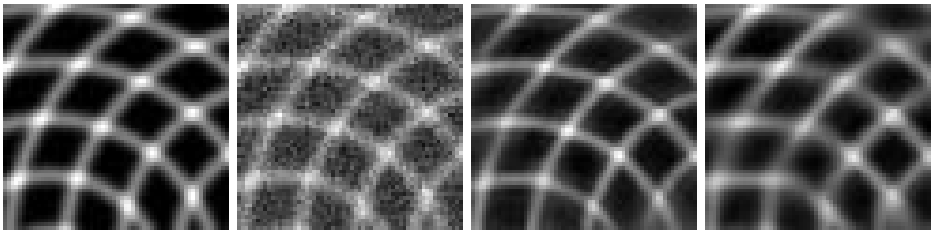
Figure 6.9 demonstrates the advantage of including curvature. Again, the same parameters and numerical scheme were used. Clearly, at $t = 30$ the circle with highest curvature is blurred if no curvature is taken into account. If curvature is taken into account, the diffusion process adheres much better to the spiral-shape of the circle in the orientation score. For the circles with smaller curvature,



(a) Original (b) +Noise (c) $t = 30$ with curvature (d) $t = 30$ no curvature

Figure 6.9: Shows the effect of including curvature on a noisy test image in CED-OS. At $t = 30$ the effect is visible: the circle with highest curvature is blurred if no curvature is taken into account. See Section 6.5 for details.

however, there is no noticeable difference.



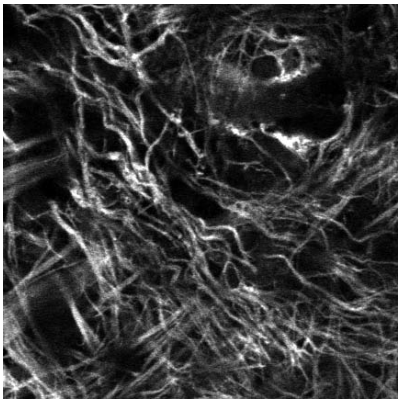
(a) Original (b) +Noise (c) $t = 24$ with d_{H} (d) $t = 24$ no d_{H}

Figure 6.10: Shows the effect of including deviation from horizontality on a noisy test image in CED-OS. At $t = 24$ the result without deviation from horizontality shows that the result is much more blurred and the lines bias towards the sampled angles $0,\pi/4$, $\pi/2$ and $3\pi/4$. If we include deviation from horizontality this problem does not occur, and even with $N_o = 4$ we are able to handle crossing correctly. See Section 6.5 for details.

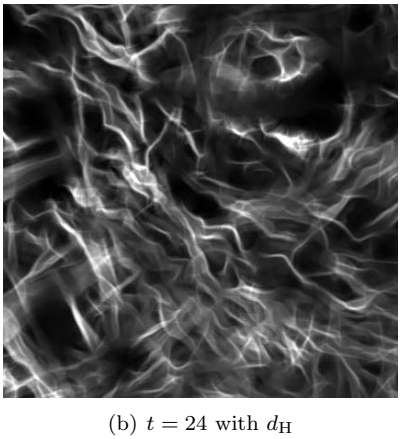
The effect of including deviation from horizontality is especially visible if we significantly lower the number of orientations N_o, because with a low number of orientations and without d_{H}, the elongated structures in the resulting images show strong biases towards the angles $l\frac{\pi}{N_o}$ with $l \in \mathbb{Z}$, while with d_{H} this problem does not occur. This is illustrated in Figure 6.10. We set $N_o = 4$, $t_s = 5$, $\rho_s = 0$, $\mu = 0.058$, and $c = 0.1$ and use the numerical scheme of Subsection 6.4.1. Clearly we observe that 4 orientations is not enough without using d_{H}, since the orientations of the curves strongly bias towards the sampled orientations. This problem is solved if we include the use of d_{H}, showing that even with only 4 orientations, we can appropriately handle crossing of two lines. Figure 6.11 shows the same effect on a microscopy image of bone tissue, where the parameters were set to $N_o = 4$, $t_s = 2$, $\rho_s = .5$, $\mu = 0.11$, and $c = 0.01$.

This means that including deviation from horizontality can make the algorithm much more efficient, since we can get good results with a very low number of orientations. Note, however, that if one wants to handle crossings of more than 2 lines or if the angle between the crossing lines is small, it is still necessary to increase the number of orientations.

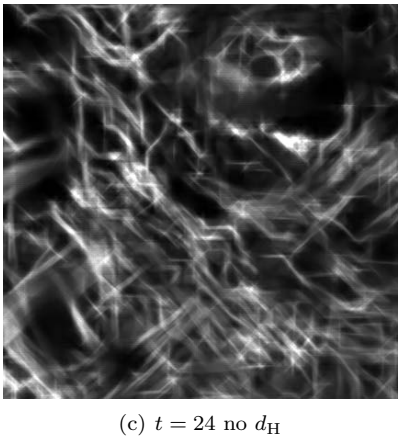
Although CED-OS is clearly advantageous for handling crossings, CED has the advantage that it is faster. In our C++ implementation, which is not fully optimized, a single iteration of CED on 128×128 roughly took 0.004 seconds while an iteration of CED-OS with the scheme of Subsection 6.4.1 and with $N_o = 32$ orientations took 0.5 seconds on a 2.4 GHz single core CPU. The large difference in computation times is caused by the fact that CED applies numerical diffusion on a 2D dataset while CED-OS applies numerical diffusion to a 3D dataset, resulting in a multiplication of algorithmic complexity with a factor N_o. Furthermore, the LSAS scheme that we used for CED is a very efficient scheme and our CPU implementation was highly optimized, while our current implementations for CED-OS



(a) Original



(b) $t = 24$ with d_{H}



(c) $t = 24$ no d_{H}

Figure 6.11: Shows the effect of including deviation from horizontality on a microscopy image of bone tissue. Clearly, with d_{H} we are able to handle crossings correctly even with $N_o = 4$. See Section 6.5 for details.

can be optimized much further.

6.6 Conclusions

In this chapter we introduced a nonlinear coherence-enhancing diffusion method on invertible orientation scores. We proposed two explicit numerical schemes to apply the nonlinear diffusion on the orientation score, and derived stability bounds for both of these schemes. The simple explicit finite difference scheme is efficient but is not optimal concerning rotational invariance and oscillations at the Nyquist frequency. The left-invariant explicit finite difference scheme with spline interpolation, on the other hand, improves the left-invariance, but is very inefficient if deviation from horizontality is included.

Especially at crossings our method renders a more natural result compared to coherence-enhancing diffusion. The diffusion shows the typical nonlinear diffusion behavior when increasing time: blurring occurs, but the important features of images are preserved over a longer range of diffusion time. Furthermore, deviation from horizontality d_{H} helps to get sharper results without orientational biases towards the sampled orientations, especially if we lower the number of orientations N_o. Therefore, including d_{H} enables a large reduction of computational demands and storage requirements.

Future research could include a feasibility study on a specific application, where further tuning of parameters and adding application-specific knowledge will lead to improved the results. Also, there is still room for improvement of the numerical schemes for (non)linear diffusion on $SE(2)$ as well as the numerical schemes in Section 7.8 for (non)linear diffusion on $SE(3)$. For instance, it is interesting to introduce an LSAS-like method [150, 149], since this method is fast and has good rotation invariance.

Acknowledgements

The following persons are acknowledged for providing realistic image data: Mirjam Rubbens (Technische Universiteit Eindhoven) for providing the source image for Figure 6.1(a), Jasper Foolen (Technische Universiteit Eindhoven) for providing the images in Figure 6.7(a) and 6.9(a), and Christopher Shaw (University of Birmingham, UK) for providing the image in Figure 6.8(a).

Everything you can imagine is real.

Pablo Picasso (1881–1973)

7

3D Orientation Scores

7.1 Introduction

The previous chapters only dealt with orientation scores of 2D images, which yielded a 3D (2 spatial coordinates and 1 orientation angle) dataset. All concepts of the previous chapters also apply to orientation scores of *3D* images, yielding a 5D or 6D (3 spatial coordinates and 2 or 3 orientation angles) dataset. The 3D case is very relevant for many (bio)medical problems, since many (bio)medical images are intrinsically 3D.

We distinguish two types of 3D orientation scores. The most common one in practice is defined as a function $u : \mathbb{R}^3 \rtimes S^2 \to \mathbb{R}$ or $\mathbb{C}$, where $\mathbb{R}^3$ is the spatial domain and $S^2 = \{\boldsymbol{x} \in \mathbb{R}^3 \,|\, \|\boldsymbol{x}\| = 1\}$ is the domain of a unit sphere. In this chapter, the domain of u is parameterized by (x, y, z, β, γ), where (β, γ) are the spherical angles that we use to parametrize the unit sphere, as depicted in Figure 7.1, i.e.

$$\boldsymbol{n}(\beta, \gamma) = \begin{pmatrix} \cos\gamma \, \sin\beta \\ \sin\gamma \, \sin\beta \\ \cos\beta \end{pmatrix} \in S^2. \tag{7.1}$$

For notational simplicity we denote $(\beta, \gamma) \in S^2$ instead of $\boldsymbol{n}(\beta, \gamma) \in S^2$ in this chapter, implicitly using relation (7.1). Figure 7.2 shows an example clarifying the structure of a 3D orientation score. The other type of 3D orientation score is defined as a function $U : \mathbb{R}^3 \rtimes SO(3) \to \mathbb{R}$ or $\mathbb{C}$, where $SO(3)$ represents the 3D rotation group, which has to be parameterized using *three* angles (Figure 7.3). The relation between these two types will be established later on in this chapter.

Some aspects of 3D orientation scores are more difficult compared to 2D orientation scores. For instance, the implementations are more complicated and much more computationally involved. Also, the math is more complicated, firstly because the rotations are not commutative, and secondly because the domain $\mathbb{R}^3 \rtimes S^2$ of the first type of 3D orientation scores introduced above is not a group but a *coset space* of the group. These additional problems were our most important reasons to first investigate 2D orientation scores.

This chapter is a first attempt to apply many concepts introduced in the previous chapters to the three-dimensional case. We will start with the description of one particular application of interest: high angular resolution diffusion imaging (HARDI). Subsequently, the theoretical part of the chapter will start. The properties of the 3D orientation group $SO(3)$ will be introduced, specifically the relation with functions on the sphere, convolution on $SO(3)$ and on the sphere, and the spherical harmonic transform. Then, we will introduce the 3D Euclidean motion group, introducing the corresponding matrix Lie algebra and Lie group and $SE(3)$-convolution. The relevant parts of the differential geometry will be described, specifically left-invariant vector fields and derivatives, exponential curves, and expressions for curvature, torsion, and deviation from horizontality. The next topic will be the diffusion on 3D orientation scores, starting with diffusion on the

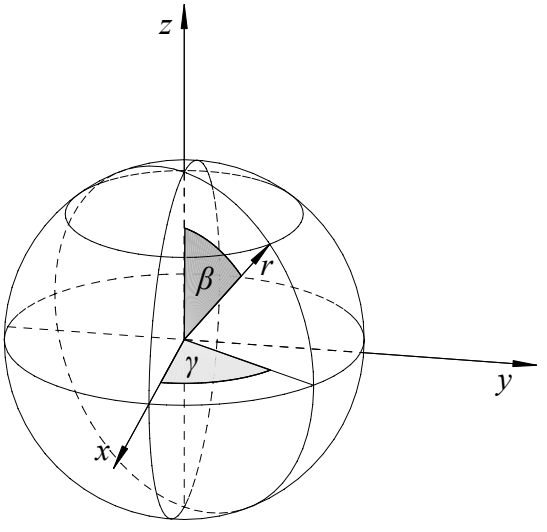


Figure 7.1: The spherical coordinate system cf. (7.1) with the azimuthal angle γ starting with 0° at the x-axis and polar angle β with 0° along the z-axis.

sphere, and subsequently diffusion on $SE(3)$ and $\mathbb{R}^3 \rtimes S^2$. Subsequently, the more practical part of the chapter starts, first discussing sampling of the sphere, and proposing implementations for different aspects of 3D orientation scores: $SE(3)$-convolution, left-invariant derivatives, and (non)linear diffusion. The chapter will end with some preliminary results on artificial HARDI datasets.

Not all methods described for 2D orientation scores will be described for 3D orientation scores in this chapter. We do not consider the problem of transforming a 3D image to an orientation score here, since the application that is of our primary interest (HARDI) does not need such a transformation, as the data coming from the MRI scanner is already a 3D orientation score. More information on invertible 3D orientation score transforms can be found in [116] and [33, Section 4.7]. Furthermore, the steerable $SE(3)$-convolution is excluded since the practical benefit is minor in the context of nonlinear diffusion on $SE(3)$, which is our primary aim. Furthermore, it is described extensively in [2, Chapter 6]. Also, steerable tensor voting in 3D is not treated since it is already described by Duits [33, Chapter 5] and later on by Reisert and Burkhardt [118], who were inspired by our work [51].

7.2 Relation with DTI and HARDI

Diffusion tensor imaging (DTI) and high angular resolution diffusion imaging (HARDI) are popular magnetic resonance imaging (MRI) techniques for imaging

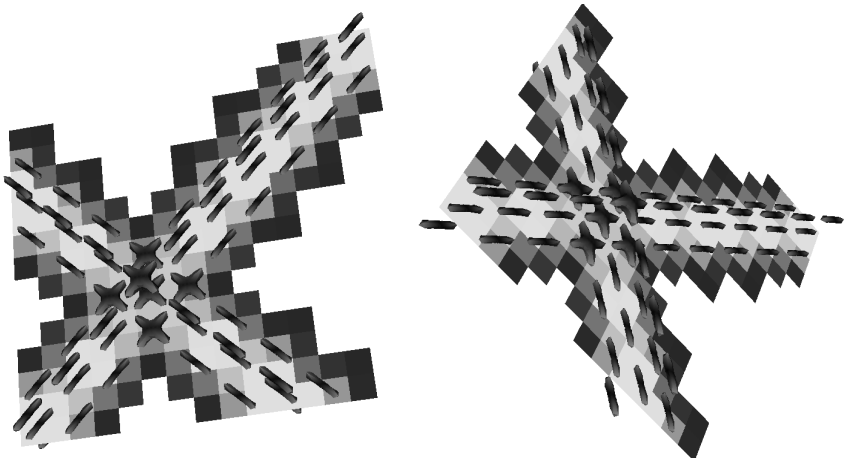


Figure 7.2: Visualization of a simple 3D orientation score $u(x, y, z, \beta, \gamma)$ containing two crossing straight lines, visualized using Q-ball glyphs in the DTI tool [32] from two different viewpoints. At each spatial position (x, y, z) a so-called *glyph* is displayed, which represents a surface in $\mathbb{R}^3$, i.e. $S^2 \to \mathbb{R}^3$. The glyph surface at each position $(x, y, z) \in \mathbb{R}^3$ is given by $(\beta, \gamma) \mapsto (x, y, z) + \mu\, u(x, y, z, \beta, \gamma)\boldsymbol{n}(\beta, \gamma)$ where $\boldsymbol{n}$ is defined in (7.1) and $\mu \in \mathbb{R}^+$ is a scaling factor determining the size of the visualized glyph, so at each orientation (β, γ) the radius of the glyph relative to its center position (x, y, z) is proportional to the response $u(x, y, z, \beta, \gamma)$.

apparent water diffusion processes in fibrous tissue, for instance white matter in the brain or muscles, such as the heart. These techniques are based on the assumption of Brownian motion of water molecules, which is expected to be larger in the direction of fibres. That is, roughly speaking the MRI scanner measures the probability of finding a water molecule at position $\boldsymbol{x}$ for a certain direction $\boldsymbol{n}(\beta, \gamma)$, where the number of acquired directions can be varied. In DTI [9] [10] the diffusivity profile at each spatial position is modeled by a rank-2 symmetric positive-definite diffusion *tensor*. With the term HARDI we refer to all diffusion MRI techniques, in which the diffusion profile on each spatial position is modeled by a *function on the sphere*, which provides richer information especially in regions where different fibrous structures cross or bifurcate. Different HARDI techniques were proposed, namely the higher order diffusion tensor model [108], the diffusion orientation transform (DOT) [109], Q-ball imaging [28], and the diffusion tensor distribution model [79]. These techniques are different from each other in the way the function on the sphere is obtained from the MRI data and the way they are represented and processed, see [113] for a comparison.

A DTI dataset consists of a positive definite 2-tensor for each spatial position, i.e. it can be regarded as a matrix-valued 3D image with a symmetric positive-definite 3×3 matrix on each spatial position. There exist a bijective mapping between the

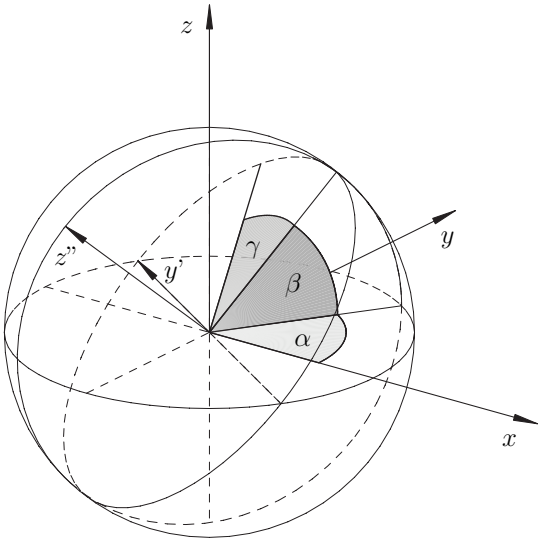


Figure 7.3: Illustration of the rotation of the x-axis using our convention (7.5) for the Euler angles (α, β, γ).

matrix representation $\mathbf{A}$ of a symmetric positive semi-definite 2-tensor $a \in T_2(\mathbb{R}^3)$ and a function on the sphere $A : S^2 \to \mathbb{R}$, given by

$$A(\beta,\gamma) = \boldsymbol{n}(\beta,\gamma)^{\mathrm{T}} \mathbf{A}\, \boldsymbol{n}(\beta,\gamma) = a(\boldsymbol{n}(\beta,\gamma), \boldsymbol{n}(\beta,\gamma)), \tag{7.2}$$

where $\boldsymbol{n}(\beta,\gamma)$ converts the spherical coordinates (β,γ) to a unit vector by using equation (7.1). As a result, a DTI image $\mathbf{D} : \mathbb{R}^3 \to T_2(\mathbb{R}^3)$ can be translated to an orientation score $u : \mathbb{R}^3 \rtimes S^2 \to \mathbb{R}$ by $u(\boldsymbol{x},\beta,\gamma) = \boldsymbol{n}(\beta,\gamma)^{\mathrm{T}} \mathbf{D}(\boldsymbol{x}) \boldsymbol{n}(\beta,\gamma)$. Notice that the property of positive semi-definiteness of $\mathbf{D}(\boldsymbol{x}) \in T_2(\mathbb{R}^3)$ translates to

$$\boldsymbol{n}(\beta,\gamma)^{\mathrm{T}} \mathbf{D}(\boldsymbol{x}) \boldsymbol{n}(\beta,\gamma) \geq 0 \Leftrightarrow u(\boldsymbol{x},\beta,\gamma) \geq 0,\ \forall \boldsymbol{x} \in \mathbb{R}^3,\ \forall (\beta,\gamma) \in S^2. \tag{7.3}$$

Clearly, all data obtained using any HARDI technique, as well as DTI data, can be considered as 3D orientation scores. In this chapter it is not of our interest what particular technique is used; we propose generally applicable techniques for smoothing and enhancing functions on $\mathbb{R}^3 \rtimes S^2$ using the structure of the 3D Euclidean motion group.

Remarkably, in HARDI processing algorithms that are proposed in literature, one usually only considers the orientational part, i.e. the data is processed as function on the sphere for each spatial position separately, see e.g. [108, 28, 48]. In tensor processing algorithms, on the other hand, one often only considers the spatial part, i.e. processing is applied to each tensor component separately [6, 17, 15]. In our approach, we consider both the spatial and the orientational part to be

included in the *domain*, so HARDI and DTI data are considered as a functions $\mathbb{R}^3 \rtimes S^2 \to \mathbb{R}$. The novelty of this work is that we define processing operations on the *entire* domain of HARDI or DTI data using the proper underlying group structure. The advantage is that we can enhance the data using both orientational and spatial neighborhood information, which potentially leads to improved enhancement and detection algorithms. Furthermore, preserving positive definiteness, which is especially a nontrivial problem in DTI processing [135, 6], becomes straightforward by constructing operations that preserve positivity. For example, a group convolution with a strictly positive kernel guarantees positivity.

7.3 The Rotation Group $SO(3)$

The noncommutative group of 3D rotations is defined as matrix group by

$$SO(3) = \{\mathbf{R} \,|\, \mathbf{R} \in \mathbb{R}^{3\times 3}, \mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1}, \det(\mathbf{R}) = 1\}. \tag{7.4}$$

In this section, we will first consider different parameterizations of $SO(3)$. Then, we will describe the coset space $SO(3)/SO(2)$, which is essential prerequisite to relate functions on the sphere (i.e. two angles) to functions on $SO(3)$ (i.e. three angles). Subsequently, we describe convolution on $SO(3)$ and on the sphere, and the spherical harmonic transform.

7.3.1 Parametrization of $SO(3)$

All 3D rotation matrices can be expressed using the three *Euler angles*, where we use the following convention, illustrated in Figure 7.3,

$$\mathbf{R}_{(\alpha,\beta,\gamma)} = \mathbf{R}_{\gamma}^{\boldsymbol{e}_z} \mathbf{R}_{\beta}^{\boldsymbol{e}_y} \mathbf{R}_{\alpha}^{\boldsymbol{e}_z}, \tag{7.5}$$

where $\mathbf{R}_{\alpha}^{\boldsymbol{n}}$ denotes rotation over α around the axis defined by vector $\boldsymbol{n}$ and $\boldsymbol{e}_y$ and $\boldsymbol{e}_z$ denote the unit vectors aligned with the y-, and z-axis respectively, i.e.

$$\mathbf{R}_{\gamma}^{\boldsymbol{e}_z} = \begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \mathbf{R}_{\beta}^{\boldsymbol{e}_y} = \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix}. \tag{7.6}$$

The Euler angles cf. (7.5) give problems at $\beta = 0$ and $\beta = \pi$ when performing differential geometry on $SO(3)$. To see why problems arise, note that

$$\mathbf{R}_{(\alpha,0,\gamma)} = \mathbf{R}_{\gamma+\alpha}^{\boldsymbol{e}_z}, \quad \text{and} \quad \mathbf{R}_{(\alpha,\pi,\gamma)} = \mathbf{R}_{\gamma-\alpha}^{\boldsymbol{e}_z} \mathbf{R}_{\pi}^{\boldsymbol{e}_y}. \tag{7.7}$$

These equations show that at $\beta = 0$ (resp. $\beta = \pi$) all values of γ and α with the same sum $\gamma + \alpha$ (resp. difference $\gamma - \alpha$) map to the same group element, making

the coordinate chart ambiguous there. To resolve this problem we introduce, where needed, a secondary coordinate chart for $SO(3)$

$$\mathbf{R}_{(\check{\alpha},\check{\beta},\check{\gamma})} = \mathbf{R}^{\boldsymbol{e}_x}_{\check{\gamma}} \mathbf{R}^{\boldsymbol{e}_y}_{\check{\beta}} \mathbf{R}^{\boldsymbol{e}_z}_{\check{\alpha}}, \quad \text{with } \mathbf{R}^{\boldsymbol{e}_x}_{\check{\gamma}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\check{\gamma} & -\sin\check{\gamma} \\ 0 & \sin\check{\gamma} & \cos\check{\gamma} \end{pmatrix}. \tag{7.8}$$

This coordinate chart is ambiguous at $\check{\beta} = \pi/2$ (since $\mathbf{R}_{(\check{\alpha},\pi/2,\check{\gamma})} = \mathbf{R}^{\boldsymbol{e}_x}_{\check{\alpha}+\check{\gamma}} \mathbf{R}^{\boldsymbol{e}_y}_{\pi/2}$) and $\check{\beta} = -\pi/2$ (since $\mathbf{R}_{(\check{\alpha},-\pi/2,\check{\gamma})} = \mathbf{R}^{\boldsymbol{e}_x}_{\check{\alpha}+\check{\gamma}} \mathbf{R}^{\boldsymbol{e}_y}_{-\pi/2}$). Thus, it should be used in combination with the other coordinate chart. Conversion between the two coordinate charts can be accomplished simply by solving the equality

$$\mathbf{R}^{\boldsymbol{e}_x}_{\check{\gamma}} \mathbf{R}^{\boldsymbol{e}_y}_{\check{\beta}} \mathbf{R}^{\boldsymbol{e}_z}_{\check{\alpha}} = \mathbf{R}^{\boldsymbol{e}_z}_{\gamma} \mathbf{R}^{\boldsymbol{e}_y}_{\beta} \mathbf{R}^{\boldsymbol{e}_z}_{\alpha}. \tag{7.9}$$

A special property of the first coordinate chart (7.5) is that it coincides with convention (7.1) for spherical coordinates when setting $\beta = \theta$ and $\gamma = \phi$ and rotating the vector $\boldsymbol{e}_z$, i.e.

$$\boldsymbol{n}(\beta,\gamma) = \mathbf{R}^{\boldsymbol{e}_z}_{\gamma} \mathbf{R}^{\boldsymbol{e}_y}_{\beta} \mathbf{R}^{\boldsymbol{e}_z}_{\alpha} \boldsymbol{e}_z = \mathbf{R}^{\boldsymbol{e}_z}_{\gamma} \mathbf{R}^{\boldsymbol{e}_y}_{\beta} \boldsymbol{e}_z. \tag{7.10}$$

On the other hand, the second coordinate chart cf. (7.8) has the convenient property that it is well-defined at the identity element $\mathbf{R} = \mathbf{I}$, which is not the case for the first coordinate chart.

7.3.2 The Left Coset Space $SO(3)/SO(2)$

Relation (7.10) between spherical coordinates of (7.1) and rotation convention (7.5) (7.10) shows that the resulting position on the sphere is independent on α, since $\mathbf{R}^{\boldsymbol{e}_z}_{\alpha} \boldsymbol{e}_z = \boldsymbol{e}_z$. This means that a function on the sphere is not equivalent to a function on the complete rotation group $SO(3)$, but rather a function on the set that partitions $SO(3)$ into *left cosets*. This will be clarified below.

A left coset $[g]_H$ of a group G with subgroup H is the set

$$[g]_H = gH = \{gh | h \in H\}, \tag{7.11}$$

for any $g \in G$. The left cosets form a partition of the group, i.e. the group is divided into disjoint cosets, and the set of all of these cosets is denoted by G/H. Two group elements $g_1 \in G$ and $g_2 \in G$ have an equivalence relation $g_1 \sim g_2$ if they belong to the same left coset, i.e. $g_1 H = g_2 H$. Multiplying both sides of the latter equality by g_1^{-1} we obtain $H = g_1^{-1} g_2 H$ which implies that the equivalence relation $g_1 \sim g_2$ holds iff $g_1^{-1} g_2 \in H$. Note that if the subgroup is not a *normal* subgroup, then cosets $[g]_H$ are not subgroups of G except for the coset $[e]_H = H$. A subgroup is normal iff $Hg = gH$ for all $g \in G$.

In the case $SO(3)/\text{stab}(\boldsymbol{e}_z)$, we have the equivalence relation $(\alpha_1,\beta_1,\gamma_1) \sim (\alpha_2,\beta_2,\gamma_2)$ iff

$$\mathbf{R}^{-1}_{(\alpha_1,\beta_1,\gamma_1)}\mathbf{R}_{(\alpha_1,\beta_1,\gamma_1)} = (\mathbf{R}^{\boldsymbol{e}_z}_{\alpha_1})^{-1}(\mathbf{R}^{\boldsymbol{e}_y}_{\beta_1})^{-1}(\mathbf{R}^{\boldsymbol{e}_z}_{\gamma_1})^{-1}\mathbf{R}^{\boldsymbol{e}_z}_{\gamma_2}\mathbf{R}^{\boldsymbol{e}_y}_{\beta_2}\mathbf{R}^{\boldsymbol{e}_z}_{\alpha_2} \in \text{stab}(\boldsymbol{e}_z). \tag{7.12}$$

where $\text{stab}(\boldsymbol{e}_z)$ denotes the subgroup of rotations around the z-axis. Note that $(\alpha_1,\beta_1,\gamma_1) \sim (\alpha_2,\beta_2,\gamma_2)$ iff $\gamma_1 = \gamma_2$ and $\beta_1 = \beta_2$. Therefore, the disjoint cosets are given by $[\mathbf{R}^{\boldsymbol{e}_z}_{\gamma}\mathbf{R}^{\boldsymbol{e}_y}_{\beta}]_{SO(2)}$, i.e. β and γ parameterize the partition of $SO(3)$ into disjoint cosets.

From now on we will write $SO(3)/SO(2)$ rather than $SO(3)/\text{stab}(\boldsymbol{e}_z)$ since $\text{stab}(\boldsymbol{e}_z) \equiv SO(2)$ under the isomorphism

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \leftrightarrow \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{7.13}$$

The cosets $SO(3)/SO(2)$ are isomorphic to the space of the unit vectors of (7.10), i.e.

$$SO(3)/SO(2) = \{[\mathbf{R}^{\boldsymbol{e}_z}_{\gamma}\mathbf{R}^{\boldsymbol{e}_y}_{\beta}]_{SO(2)} | (\beta,\gamma) \in S^2\} \cong S^2 = \{\boldsymbol{x} \in \mathbb{R}^3 \,|\, \|\boldsymbol{x}\| = 1\}. \tag{7.14}$$

Note that the set of all disjoint cosets $SO(3)/SO(2)$ does *not* form a group since $SO(2)$ is not a normal subgroup of $SO(2)$, so $[g_1]_{SO(2)}\,[g_2]_{SO(2)} \neq [g_1 g_2]_{SO(2)}$.

7.3.3 Functions on $SO(3)$ and S^2

In this chapter, functions $F : SO(3) \to \mathbb{C}$ will frequently occur. We will parameterize $SO(3)$ using matrices rather than Euler angles, e.g. $F(\mathbf{R})$. Equation (7.5) and (7.8) describe the conversion of two different conventions of Euler angles to rotation matrices.

A special class of functions on $SO(3)$ are *α-right-invariant* functions [1], are *independent* on Euler angle α. This means that they have the following property

$$\begin{aligned} &\tilde{F}(\mathbf{R}_{(\beta,\gamma,\alpha'+\alpha)}) = \tilde{F}(\mathbf{R}_{(\beta,\gamma,\alpha)}), \quad \text{for all } \alpha', \text{ that is,} \\ &\qquad \mathcal{Q}_{\mathbf{R}^{\boldsymbol{e}_z}_{\alpha'}} \circ \tilde{F} = \tilde{F}, \end{aligned} \tag{7.15}$$

where we denote $\tilde{F}$ instead of F to make the α-right-invariance explicitly visible in the notation. $\mathcal{Q}$ denotes the right-regular representation, defined as $(\mathcal{Q}_g \circ \tilde{F})(h) =$

[1] Note that in this context we truly mean *in*variance, whereas in case of left- and right-invariance we actually mean *co*variance. The term is covariance is not used, however, because we adhere to standard terminology in mathematics.

$\tilde{F}(h\,g)$. If F is α-right-invariant, such a function can be identified one-to-one to a function on $SO(3)/SO(2) \cong S^2$ by

$$f(\beta,\gamma) = \tilde{F}(\mathbf{R}_{(0,\beta,\gamma)}) \quad \text{iff } F \text{ is } \alpha\text{-right-invariant.} \tag{7.16}$$

In the theoretical part of this chapter we will mostly work with the α-right-invariant function $\tilde{F}$, while in the implementational part of this chapter, it will turn out to be more straightforward to use the corresponding function $f$on S^2.

7.3.4 Convolution on $SO(3)$ and S^2

In this subsection we will derive the convolutions on $SO(3)$ and S^2, see also [31]. Since $SO(3)$ is a group, convolution on $SO(3)$ is straightforward and follows directly from the general convolution equation (2.52) on page 30, i.e.

$$(\Psi_{SO(3)} *_{SO(3)} F)(\mathbf{R}) = \int\limits_{SO(3)} \Psi_{SO(3)}((\mathbf{R}')^{-1}\cdot\mathbf{R})F(\mathbf{R}')\mathrm{d}\mu(\mathbf{R}'), \tag{7.17}$$

where $\Psi_{SO(3)}$ and $\Psi_{SO(3)}$ can be any function $SO(3)\to\mathbb{C}$ and and where $d\mu(\mathbf{R}')$ denotes the Haar measure on $SO(3)$, which fullfills [65]

$$\int\limits_{SO(3)} F(\mathbf{R}')\mathrm{d}\mu(\mathbf{R}') = \int\limits_{SO(3)} F(\mathbf{R}^{-1}\cdot\mathbf{R}')\mathrm{d}\mu(\mathbf{R}'), \quad \text{for all } \mathbf{R}\in SO(3). \tag{7.18}$$

Expressed in Euler angles the Haar measure is given by

$$\int\limits_{SO(3)} F(\mathbf{R}')\mathrm{d}\mu(\mathbf{R}') = \int_0^{2\pi}\int_0^{\pi}\int_0^{2\pi} F(\mathbf{R}_{(\alpha',\beta',\gamma')})\,\sin\beta'\,\mathrm{d}\alpha'\,\mathrm{d}\beta'\,\mathrm{d}\gamma'. \tag{7.19}$$

Next, we consider convolution of functions on S^2, or equivalently, α-right-invariant functions on $SO(3)$. We require that if the input function $\tilde{F}: SO(3)\to\mathbb{R}$ is α-right-invariant, then the output of the convolution must be α-right-invariant as well. This is only guaranteed if the kernel $\tilde{\Psi}_{SO(3)}(\tilde{F}: SO(3)\to\mathbb{R})$ is independent on *both* Euler angle α and Euler angle γ, i.e.

$$\begin{aligned}\tilde{\Psi}_{SO(3)}(\mathbf{R}_{(\alpha,\beta,\gamma)}) &= \tilde{\Psi}_{SO(3)}(\mathbf{R}^{\boldsymbol{e}_z}_{\gamma'}\cdot\mathbf{R}_{(\alpha,\beta,\gamma)}) = \tilde{\Psi}_{SO(3)}(\mathbf{R}_{(\alpha,\beta,\gamma)}\cdot\mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})\\ &= \tilde{\Psi}_{SO(3)}(\mathbf{R}_{(0,\beta,0)}), \quad \text{for all } \alpha,\ \beta,\ \gamma,\ \alpha', \text{ and } \gamma'.\end{aligned} \tag{7.20}$$

This requirement on $\tilde{\Psi}_{SO(3)}$ can be easily checked by rewriting expression $\Psi_{SO(3)}((\mathbf{R}')^{-1}\cdot\mathbf{R})$ in (7.17) explicitly using Euler angles $\mathbf{R}=\mathbf{R}_{(\alpha,\beta,\gamma)}$ and $\mathbf{R}'=\mathbf{R}_{(\alpha',\beta',\gamma')}$, yielding

$$\Psi_{SO(3)}((\mathbf{R}')^{-1}\cdot\mathbf{R}) = \Psi_{SO(3)}((\mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})^{-1}(\mathbf{R}^{\boldsymbol{e}_y}_{\beta'})^{-1}(\mathbf{R}^{\boldsymbol{e}_z}_{\gamma'})^{-1}\mathbf{R}^{\boldsymbol{e}_z}_{\gamma}\mathbf{R}^{\boldsymbol{e}_y}_{\beta}\mathbf{R}^{\boldsymbol{e}_z}_{\alpha}). \tag{7.21}$$

The α-right-invariant input function $\tilde{F}$ in (7.17) is independent on α', so $\tilde{\Psi}_{SO(3)}$ must be independent on α' as well. Since the α-dependent rotation appears at the left-side in (7.21), this means that $(\alpha'', \beta'', \gamma'') \mapsto \Psi_{SO(3)}(\mathbf{R}_{(\alpha'',\beta'',\gamma'')})$ must be independent on γ''. Furthermore, the α-right-invariant output is indepdendent on α, which appears at the right-side in (7.21), so $\tilde{\Psi}_{SO(3)}$ must be independent on α'' as well.

For functions on the sphere $f : S^2 \to \mathbb{R}$, the convolution can thus be written as

$$\begin{aligned}&(\psi_{S^2} *_{SO(3)} f)(\beta, \gamma)\\ &\quad = \int_0^{2\pi}\int_0^{\pi} \psi_{S^2}(\beta[(\mathbf{R}_{\beta'}^{\boldsymbol{e}_y})^{-1}(\mathbf{R}_{\gamma'}^{\boldsymbol{e}_z})^{-1}\mathbf{R}_{\gamma}^{\boldsymbol{e}_z}\mathbf{R}_{\beta}^{\boldsymbol{e}_y}])f(\beta', \gamma') \sin\beta' \,\mathrm{d}\beta' \,\mathrm{d}\gamma',\end{aligned} \tag{7.22}$$

where $\psi_{S^2}(\beta) = \tilde{\Psi}_{SO(3)}(\mathbf{R}_{(0,\beta,0)})$ and $\beta[\mathbf{R}]$ obtains Euler angle β from rotation matrix $\mathbf{R} \in SO(3)$.

7.3.5 The Spherical Harmonic Transform

One can define the Fourier transform on $SO(3)$ cf. [22]. In this work, we are only interested in a specific case, namely the Fourier transform on the sphere S^2, which we will introduce in this subsection. The *spherical harmonic transform* (SHT) plays the role of the Fourier transform on the sphere. The SHT of a function $f : S^2 \to \mathbb{C}$ is defined by

$$\mathcal{F}_{S^2}[f]_m^l = \int_0^{2\pi}\int_0^{\pi} f(\beta, \gamma)\overline{Y_m^l}(\beta, \gamma) \sin\beta \mathrm{d}\beta\mathrm{d}\gamma, \tag{7.23}$$

where Y_m^l denote the spherical harmonics, and l and m enumerate the spherical harmonics where $l \in \mathbb{N} \cup \{0\}$ and $m \in \{-l, -l+1, \ldots, l-1, l\}$ (where $Y_m^l = \overline{Y_{-m}^l}$),

$$Y_m^l(\beta, \gamma) = \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}} e^{im\gamma} P_m^l(\cos\beta), \tag{7.24}$$

where P_m^l are the associated Legendre polynomials

$$P_m^l(z) = \frac{(-1)^m}{2^l\, l!}(1-z^2)^{\frac{m}{2}} \frac{\mathrm{d}^{l+m}}{\mathrm{d}z^{l+m}}(z^2-l)^l, \tag{7.25}$$

with $-1 \leq z \leq 1$. An essential property of the spherical harmonics is orthogonality

$$\int_0^{2\pi}\int_0^{\pi} Y_{m_1}^{l_1}(\beta, \gamma)\overline{Y_{m_2}^{l_2}(\beta, \gamma)}\mathrm{d}\gamma \sin\beta \mathrm{d}\beta = \delta_{l_1,l_2}\delta_{m_1,m_2}. \tag{7.26}$$

The index l enumerates the invariant subspaces under rotation, i.e. the rotation of a spherical harmonic Y_m^l is given by

$$\mathcal{R}_{(\alpha',\beta',\gamma')}[Y_m^l](\beta,\gamma) = \sum_{m'=-l}^{l} \overline{\Gamma^l_{m'\,m}}(\alpha',\beta',\gamma') Y^l_{m'}(\beta,\gamma), \tag{7.27}$$

where $\Gamma^l_{m'\,m}$ denotes the Wigner's D-functions, which are defined in for instance [2, Chapter 6, page 167]. The Wigner's D-functions are the irreducible representations of $SO(3)$.

The SHT is exactly invertible and the inverse SHT of $\hat{f} = \mathcal{F}_{S^2}[f]$ is given by

$$f = \mathcal{F}^{-1}_{S^2}[\hat{f}](\beta,\gamma) = \sum_{l=-\infty}^{\infty} \sum_{m=-l}^{l} \hat{f}^l_m Y^l_m(\beta,\gamma). \tag{7.28}$$

Since ψ_{S^2} in (7.22) is independent on β only, the spherical harmonic transform of ψ_{S^2} has the property (cf. the definition of spherical harmonics (7.24))

$$\mathcal{F}_{S^2}[\psi_{S^2}]^l_m = 0, \text{ for all } m \neq 0. \tag{7.29}$$

Therefore, the convolution theorem for S^2-convolution (7.22) becomes

$$\mathcal{F}_{S^2}[\psi_{S^2} *_{S^2} f]^l_m = \mathcal{F}_{S^2}[\psi_{S^2}]^l_0 \, \mathcal{F}_{S^2}[f]^l_m. \tag{7.30}$$

7.3.5.1 Spherical Harmonic Transform on 2-Tensors

On 2-tensors we can calculate the spherical harmonics transformation directly, similar to the 2D case described in Section 4.2. We define the following basis for symmetric 3D 2-tensors, such that the basis vectors form invariant subspaces under rotation for $l = 0, 2$ (i.e. a 1D invariant subspace for $l = 0$ and a 5D invariant subspace for $l = 2$).

$$\begin{aligned}
\mathbf{E}^0_0 &= \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, & \mathbf{E}^2_0 &= \frac{1}{\sqrt{6}} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}, \\
\mathbf{E}^2_{-1} &= \frac{1}{2} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & -i \\ 1 & -i & 0 \end{pmatrix}, & \mathbf{E}^2_1 &= \frac{1}{2} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & i \\ 1 & i & 0 \end{pmatrix}, \\
\mathbf{E}^2_{-2} &= \frac{1}{2} \begin{pmatrix} 1 & -i & 0 \\ -i & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & \mathbf{E}^2_2 &= \frac{1}{2} \begin{pmatrix} 1 & i & 0 \\ i & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix},
\end{aligned} \tag{7.31}$$

where $\mathbf{E}^l_m$ is an orthonormal basis with respect to the inner product $(\mathbf{A}, \mathbf{B}) = \mathrm{tr}(\mathbf{A}, \overline{\mathbf{B}}^{\mathrm{T}})$. The SHT of a tensor $\mathbf{A}$ can be found by

$$\mathcal{F}_{S^2}[\mathbf{A}]^l_m = (\mathbf{A}, \mathbf{E}^l_m). \tag{7.32}$$

7.4 The 3D Euclidean Motion Group $SE(3)$

The *3D Euclidean motion group* is the group of 3D translations and 3D rotations, i.e. $SE(3) = \mathbb{R}^3 \rtimes SO(3)$. An element of $SE(3)$ can be parameterized by $(\boldsymbol{x}, \mathbf{R})$ where $\boldsymbol{x} \in \mathbb{R}^3$ is the translation vector and $\mathbf{R} \in SO(3)$ is the rotation matrix.

The group product and inverse of $SE(3)$ are given by

$$\begin{aligned} g\, g' &= (\boldsymbol{x}, \mathbf{R})\,(\boldsymbol{x}', \mathbf{R}') = (\boldsymbol{x} + \mathbf{R} \cdot \boldsymbol{x}', \mathbf{R} \cdot \mathbf{R}'), \\ g^{-1} &= (\boldsymbol{x}, \mathbf{R})^{-1} = (-\mathbf{R}]^{-1}\boldsymbol{x}, \mathbf{R}^{-1}), \end{aligned} \tag{7.33}$$

where the rotation part is defined in Section 7.3. The Lie algebra of $SE(3)$ is spanned by

$$T_e(SE(3)) = \text{span}\{\partial_x\big|_e, \partial_y\big|_e, \partial_z\big|_e, \partial_{\check{\gamma}}\big|_e, \partial_{\check{\beta}}\big|_e, \partial_{\check{\alpha}}\big|_e\}, \tag{7.34}$$

where it should be noted that we use the second coordinate chart cf. (7.8) since the first coordinate chart cf. (7.5) is not defined at the identity element e.

7.4.1 Representations

In Subsection 2.6.2 we described the most important representations for $SE(2)$, namely on $\mathbb{L}_2(\mathbb{R}^2)$ and $\mathbb{L}_2(SE(2))$. For $SE(3)$ their structure is the same, so the left- and right-regular representations on a function $F \in \mathbb{L}_2(SE(3))$ are

$$(\mathcal{L}_g \circ U)(h) \quad = (U \circ L_g^{-1})(h) = U(g^{-1}h), \quad g, h \in SE(3),, \tag{7.35}$$

$$(\mathcal{Q}_g \circ U)(h) \quad = (U \circ Q_g)(h) = U(h\, g), \quad g, h \in SE(3)., \tag{7.36}$$

The $SE(3)$ representation on 3D images $L \in \mathbb{L}_2(\mathbb{R}^3)$ is given by

$$(\mathcal{U}_g \circ L)(\boldsymbol{y}) = L(\mathbf{R}^{-1}(\boldsymbol{y} - \boldsymbol{x})),\ L \in \mathbb{L}_2(\mathbb{R}^3),\ g = (\boldsymbol{x}, \mathbf{R}) \in SE(3),\ \boldsymbol{y} \in \mathbb{R}^3. \tag{7.37}$$

7.4.2 Matrix Lie Algebra and Group

The matrix Lie algebra $T_e(SE(3))$ is spanned by the following basis

$$\begin{aligned}
\mathbf{X}_1 &= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{X}_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \\
\mathbf{X}_2 &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{X}_5 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \\
\mathbf{X}_3 &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \qquad \mathbf{X}_6 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.
\end{aligned} \tag{7.38}$$

Consequently, the commutators are given by

$$[\mathbf{X}_i, \mathbf{X}_j] = \mathbf{X}_i\mathbf{X}_j - \mathbf{X}_j\mathbf{X}_i = \begin{pmatrix} 0 & 0 & 0 & 0 & \mathbf{X}_3 & -\mathbf{X}_2 \\ 0 & 0 & 0 & -\mathbf{X}_3 & 0 & \mathbf{X}_1 \\ 0 & 0 & 0 & \mathbf{X}_2 & -\mathbf{X}_1 & 0 \\ 0 & \mathbf{X}_3 & -\mathbf{X}_2 & 0 & \mathbf{X}_6 & -\mathbf{X}_5 \\ -\mathbf{X}_3 & 0 & \mathbf{X}_1 & -\mathbf{X}_6 & 0 & \mathbf{X}_4 \\ \mathbf{X}_2 & -\mathbf{X}_1 & 0 & \mathbf{X}_5 & -\mathbf{X}_4 & 0 \end{pmatrix}, \tag{7.39}$$

where i enumerates vertically and j horizontally.

By calculating the matrix exponents we find the following matrix representation of the $SE(3)$ group

$$\begin{aligned}
\mathbf{E}_{(\boldsymbol{x},\mathbf{R})} &= \exp(x\,\mathbf{X}_1 + y\,\mathbf{X}_2 + z\,\mathbf{X}_3)\,\exp(\check{\gamma}\,\mathbf{X}_4)\,\exp(\check{\beta}\,\mathbf{X}_5)\,\exp(\check{\alpha}\,\mathbf{X}_6) \\
&= \begin{pmatrix} \mathbf{R} & \boldsymbol{x} \\ \mathbf{0} & 1 \end{pmatrix}, \qquad \text{with } \mathbf{R} = \mathbf{R}_{\check{\gamma}}^{\boldsymbol{e}_x}\mathbf{R}_{\check{\beta}}^{\boldsymbol{e}_y}\mathbf{R}_{\check{\alpha}}^{\boldsymbol{e}_z}.
\end{aligned} \tag{7.40}$$

Note that the angles $(\check{\alpha}, \check{\beta}, \check{\gamma})$ adhere to the second spherical coordinate chart cf. (7.8).

7.4.3 Left- and Right-Invariances

Remember from Section 2.7 the notion of left-invariance of an operator Φ

$$\forall g \in SE(2) : \mathcal{L}_g \circ \Phi = \Phi \circ \mathcal{L}_g, \tag{7.41}$$

and similarly right-invariance

$$\forall g \in SE(2) : \mathcal{Q}_g \circ \Phi = \Phi \circ \mathcal{Q}_g. \tag{7.42}$$

We showed that for 2D orientation scores constructed from 2D images, operators are required to be left-invariant for rotation-invariance on the image, while right-invariance is not desirable. These requirements were derived by considering the wavelet transform that is used to construct 2D orientation scores from 2D images. Since we consider applications that directly yield orientation scores (HARDI and DTI), these requirements might seem void in this context.

However, in the current case we propose to keep the same requirements on left- and right-invariance. The reason for this will be clarified below. Define $\mathcal{W} : (SE(3) \to \mathbb{C}) \to (\mathbb{R}^3 \to \mathbb{C})$ to be the operator that calculates the *orientation-marginal*,

$$\mathcal{W}[U](\boldsymbol{x}) = \int_{SO(3)} U(\boldsymbol{x}, \mathbf{R}) \mathrm{d}\mu(\mathbf{R}). \tag{7.43}$$

It is easy to derive that

$$\mathcal{U}_g \circ \mathcal{W} \circ U = \mathcal{W} \circ \mathcal{L}_g \circ U, \ \forall\, g \in SE(3), \tag{7.44}$$

where $(\mathcal{U}_{(\boldsymbol{x}', \mathbf{R}')} f)(\boldsymbol{x}) = f((\mathbf{R}')^{-1}(\boldsymbol{x} - \boldsymbol{x}'))$, while

$$\tilde{\mathcal{U}}_g \circ \mathcal{W} \circ U \neq \mathcal{W} \circ \mathcal{Q}_g \circ U, \tag{7.45}$$

for all $g \in SE(3)$ and any definition of $\tilde{\mathcal{U}}_g$. In words, the left-regular representation "commutes" with $\mathcal{W}$, where $\mathcal{L}_g$ changes into $\mathcal{U}_g$ since the function space changes, while it is not possible to find such a relation for the right-regular representation. This becomes apparent in the following equation,

$$(\mathcal{W} \circ \mathcal{Q}_{(\boldsymbol{x}, \mathbf{R})} \circ U)(\boldsymbol{x}', \mathbf{R}') = \int_{SO(3)} U(\boldsymbol{x}' + \mathbf{R}'\boldsymbol{x}, \mathbf{R}'\,\mathbf{R}) \mathrm{d}\mu(\mathbf{R}'), \tag{7.46}$$

which shows that the integral variable $\mathbf{R}'$ enters the spatial part, making it impossible to first apply $\mathcal{W}$ and then some operation $\tilde{\mathcal{U}}_g$ to yield the same result for all $g \in SE(3)$. This observation makes it sensible to favor operators Φ to be left-invariant, i.e.

$$\mathcal{W} \circ \Phi \circ \mathcal{L}_g \circ U = \mathcal{W} \circ \mathcal{L}_g \circ \Phi \circ U = \mathcal{U}_g \circ \mathcal{W} \circ \Phi \circ U, \tag{7.47}$$

states that applying a group transformation ($\mathcal{L}_g$) on the input U renders the same result as applying the same group transformation ($\mathcal{U}_g$) on the orientation-marginal of the output. Therefore, we still restrict to operators Φ that are left-invariant and not right-invariant.

7.4.4 Functions on $SE(3)$ and $\mathbb{R}^3 \rtimes S^2$

In the beginning of this chapter (Section 7.1), we defined a 3D orientation score u as a function of three spatial variables and only two angular variables describing a position on the sphere. In Subsection 7.4.1, we introduced a more general function U on $SE(3)$, i.e. a function on three spatial variables and *three* angles. The latter function U is considered as a *general orientation score*, which in practice is not likely to occur, but it is important for the theory. Analogously to Subsection 7.3.2, in which we described why spherical coordinates do not constitute a group, a 3D orientation score u cf. Section 7.1 is not a function on the group, but rather a function on the coset space $SE(3)/(\mathbf{0}\times\text{stab}(\boldsymbol{e}_z))$. Here, $(\mathbf{0}\times\text{stab}(\boldsymbol{e}_z))$ denotes the $SE(2)$ subgroup of rotations around the z-axis and translation $\mathbf{0}$, i.e. in matrix form

$$(\mathbf{0} \times \text{stab}(\boldsymbol{e}_z)) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{7.48}$$

which is isomorphic to $SO(2)$. For notational convenience, in the rest of the chapter we will denote this coset space $SE(3)/(\mathbf{0} \times \text{stab}(\boldsymbol{e}_z))$ as $\mathbb{R}^3 \rtimes S^2$.

Analogously to Subsection 7.3.3, a function $\tilde{U} : SE(3) \to \mathbb{C}$ is *α-right-invariant* if

$$\begin{aligned} \mathcal{Q}_{(\mathbf{0},\mathbf{R}_{\alpha'}^{\boldsymbol{e}_z})} \circ \tilde{U} &= \tilde{U}, \quad \text{for all } \alpha', \text{ that is,} \\ \tilde{U}(\boldsymbol{x}, \mathbf{R}_{(\alpha'+\alpha,\beta,\gamma)}) &= \tilde{U}(\boldsymbol{x}, \mathbf{R}_{(\alpha,\beta,\gamma)}), \end{aligned} \tag{7.49}$$

where we write $\tilde{U}$ rather than U in the rest of the chapter to make explicit in the notation that the function is α-right-invariant. Since $\tilde{U}$ is independent on angle α, $\tilde{U}$ can be identified one-to-one to an orientation score $u : \mathbb{R}^3 \rtimes S^2 \to \mathbb{C}$, as

$$u(\boldsymbol{x}, \beta, \gamma) = \tilde{U}(\boldsymbol{x}, \mathbf{R}_{(0,\beta,\gamma)}) \quad \text{where } \tilde{U} \text{ is } \alpha\text{-right-invariant.} \tag{7.50}$$

In the theoretical part of this chapter we will mostly work with the α-right-invariant function $\tilde{U}$, because it is more convenient to work with functions on the group. In the implementation part of this chapter, on the other hand, it will turn out to be more straightforward to use the orientation score $f$on $\mathbb{R}^3 \rtimes S^2$.

7.4.5 $SE(3)$-Convolutions

The equation for the group convolution of $SE(3)$ is

$$(\Psi *_{SE(3)} U)(g) = \int_{SE(3)} \Psi(h^{-1}g) U(h) dh. \tag{7.51}$$

This yields

$$(\Psi *_{SE(3)} U)(\boldsymbol{x}, \mathbf{R}) = \int_{\mathbb{R}^3} \int_{SO(3)} \Psi(\mathbf{R}'^{-1}(\boldsymbol{x} - \boldsymbol{x}'), \mathbf{R}'^{-1}\mathbf{R}) U(\boldsymbol{x}', \mathbf{R}') \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\mu(\mathbf{R}'), \tag{7.52}$$

where $\mathrm{d}\mu(\mathbf{R}')$ is defined in (7.19).

For an α-right-invariant $\tilde{U}$ cf. (7.49) we need to put additional requirements on the kernel Ψ, similar to Subsection 7.3.4 for convolution on the sphere. We require the result $\Psi *_{SE(3)} \tilde{U}$ to be α-right-invariant as well, leading to the following requirement

$$\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})} \circ (\tilde{\Psi} *_{SE(3)} (\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha''})} \circ \tilde{U})) = \tilde{\Psi} *_{SE(3)} \tilde{U}, \tag{7.53}$$

which imposes requirements on the kernel $\tilde{\Psi}$. Remember the properties of $SE(2)$-convolution (3.23) and (3.24) on page 53, which also apply to $SE(3)$, i.e.

$$\mathcal{Q}_g(\Psi *_{SE(3)} U) = (\mathcal{Q}_g \Psi) *_{SE(3)} U, \ \forall g \in SE(3), \tag{7.54}$$

$$(\mathcal{L}_g \Psi) *_{SE(3)} U = \Psi *_{SE(3)} (\mathcal{Q}_{g^{-1}} U), \ \forall g \in SE(3). \tag{7.55}$$

The left side of (7.53) can now be rewritten as

$$\begin{aligned} &\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\tilde{\alpha}})} \circ (\tilde{\Psi} *_{SE(3)} (\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha''})} \circ \tilde{U})) \\ &\overset{\text{Eq.}(7.54)}{=} ((\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})} \circ \tilde{U}) *_{SE(3)} (\mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\tilde{\alpha}''})} \circ \tilde{U})) \\ &\overset{\text{Eq.}(7.55)}{=} (\mathcal{L}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{-\alpha''})} \circ \mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})} \circ \tilde{\Psi}) *_{SE(3)} \tilde{U}. \end{aligned} \tag{7.56}$$

This yields

$$\tilde{\Psi} = \mathcal{L}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{-\alpha''})} \circ \mathcal{Q}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\tilde{\alpha}})} \circ \tilde{\Psi}, \ \text{for all } \alpha', \ \alpha'', \tag{7.57}$$

so $\tilde{\Psi}$ is required to be both α-right-invariant and α-left-invariant (i.e. $\mathcal{L}_{(\mathbf{0}, \mathbf{R}^{\boldsymbol{e}_z}_{\alpha'})} \circ \tilde{U} = \tilde{U}$ for all α'). More explicitly this yields

$$\tilde{\Psi}(\boldsymbol{x}, \mathbf{R}^{\boldsymbol{e}_z}_{\gamma} \mathbf{R}^{\boldsymbol{e}_y}_{\beta} \mathbf{R}^{\boldsymbol{e}_z}_{\alpha}) = \tilde{\Psi}((\mathbf{R}^{\boldsymbol{e}_z}_{\alpha''})^{-1} \boldsymbol{x}, \mathbf{R}^{\boldsymbol{e}_z}_{\gamma - \alpha''} \mathbf{R}^{\boldsymbol{e}_y}_{\beta} \mathbf{R}^{\boldsymbol{e}_z}_{\alpha + \alpha'}), \ \text{for all } \alpha', \ \alpha''. \tag{7.58}$$

Using the results above we can formulate the $\mathbb{R}^3 \rtimes S^2$-convolution as follows

$$\begin{aligned} &(\psi *_{\mathbb{R}^3 \rtimes S^2} u)(\boldsymbol{x}, \beta, \gamma) \\ &= \int_{\mathbb{R}^3} \int_0^{2\pi} \int_0^{\pi} \psi((\mathbf{R}^{\boldsymbol{e}_z}_{\gamma'})^{-1} (\mathbf{R}^{\boldsymbol{e}_y}_{\beta'})^{-1} \boldsymbol{x}, \boldsymbol{\varpi}_{S^2}((\mathbf{R}^{\boldsymbol{e}_y}_{\beta'})^{-1} (\mathbf{R}^{\boldsymbol{e}_z}_{\gamma'})^{-1} \mathbf{R}^{\boldsymbol{e}_z}_{\gamma} \mathbf{R}^{\boldsymbol{e}_y}_{\beta})) \\ &\qquad \times u(\boldsymbol{x}', \beta', \gamma') \mathrm{d}\boldsymbol{x}' \, \sin\beta' \, \mathrm{d}\beta' \, \mathrm{d}\gamma', \end{aligned} \tag{7.59}$$

where $\boldsymbol{\varpi}_{S^2}$ renders the Euler angles (β, γ) corresponding to the rotation matrix and where ψ and $\tilde{U}$ are given by

$$\begin{aligned} &\psi(\boldsymbol{x}, \beta, \gamma) = \tilde{\Psi}(\boldsymbol{x}, \mathbf{R}_{\gamma}^{\boldsymbol{e}_z} \mathbf{R}_{\beta}^{\boldsymbol{e}_y}), \ \tilde{\Psi} \text{ is } \alpha\text{-right-invariant and } \alpha\text{-left-invariant,} \\ &\tilde{U}(\boldsymbol{x}, \beta, \gamma) = \tilde{U}(\boldsymbol{x}, \mathbf{R}_{\gamma}^{\boldsymbol{e}_z} \mathbf{R}_{\beta}^{\boldsymbol{e}_y}), \ \tilde{U} \text{ is } \alpha\text{-right-invariant.} \end{aligned} \tag{7.60}$$

Note that (7.58) shows that, due to the rotation over α'' in the spatial part, the kernel ψ as a rotational symmetry around the z-axis and can therefore be expressed in the following form

$$\psi(\boldsymbol{x}, \beta, \gamma) = \psi_{\text{sub}}(\sqrt{x^2 + y^2}, z, \beta, \gamma). \tag{7.61}$$

7.4.6 Obtaining $SE(3)$-Kernels from $SE(2)$-Kernels

The requirement on ψ of (7.61), which states that the *spatial part* of the kernel must be symmetric around the z-axis, provides a simple "ad hoc" recipe to "convert" an $SE(2)$-kernel to an $SE(3)$-kernel. We simply add the requirement that the *orientational part* of the kernel is also symmetric around the z-axis, which removes the dependency on γ. Using this extra requirement, we can make a straightforward conversion from 2D to 3D by setting (illustrated in Figure 7.4)

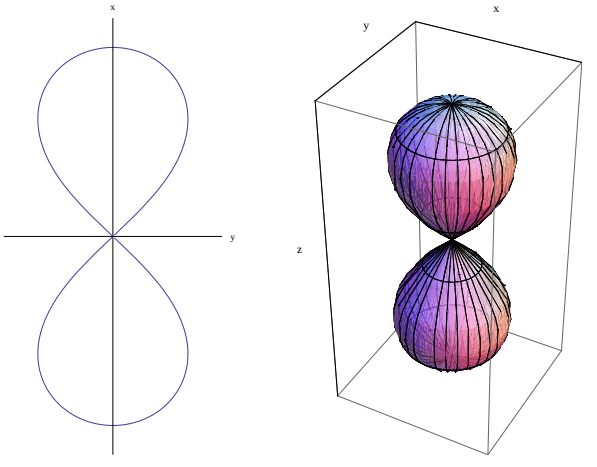
$$\psi(x, y, z, \beta, \gamma) = \Psi_{SE(2)}(z, \sqrt{x^2 + y^2}, \beta), \tag{7.62}$$

where $\Psi_{SE(2)}$ denotes any $SE(2)$-convolution kernel. Notice that the z-coordinate in 3D plays the same role as the x-coordinate in 2D, which makes sense since in our convention, a zero rotation in 2D coincides with the x-axis while a zero rotation in 3D coincides with the z-axis.

This symmetry consideration coincides with the way Medioni et al. [98] convert a 2D tensor voting field to a 3D voting field.

7.5 Differential Geometry on $SE(3)$

In Section 2.8 we introduced the basic differential geometry on $SE(2)$. In this section we establish the same concepts for $SE(3)$. We will introduce the left-invariant vector fields, the commutator relations, the inner product and norm on the tangent spaces $T_g(SE(3))$, the exponential curves, and finally we introduce expressions for curvature, torsion, and deviation from horizontality.



(a) $SE(2)$-kernel (b) Obtained $SE(3)$-kernel

Figure 7.4: Visualization of obtaining an $SE(3)$-kernel from an $SE(2)$-kernel. Left: example iso-line of an $SE(2)$-kernel, showing the spatial axes only. Right: by assuming a symmetry around the z-axis, the $SE(2)$-kernel is converted to a $SE(3)$-kernel, shown as the iso-surface corresponding to the iso-line of (a), where again only the spatial axes are shown.

7.5.1 Left-Invariant Vector Fields in $SE(3)$

Using the matrix representation cf. equation (7.40), left-invariant vector fields are given by

$$(\mathcal{A}_i U)(\mathbf{E}_g) = \lim_{h \to 0} \frac{U(\mathbf{E}_g \cdot \exp(h\,\mathbf{X}_i)) - U(\mathbf{E}_g)}{h}. \tag{7.63}$$

Expressed in the *first coordinate chart* of eq. (7.5) this renders for the left-invariant derivatives at position $g = (x, y, z, \mathbf{R}_{(\alpha,\beta,\gamma)})$ (following [22, Section 9.10])

$$\begin{aligned}
\mathcal{A}_1\big|_g &= (\cos\alpha\cos\beta\cos\gamma - \sin\alpha\sin\gamma)\,\partial_x \\
&\quad + (\sin\alpha\cos\gamma + \cos\alpha\cos\beta\sin\gamma)\,\partial_y - \cos\alpha\sin\beta\,\partial_z, \\
\mathcal{A}_2\big|_g &= (-\sin\alpha\cos\beta\cos\gamma - \cos\alpha\sin\gamma)\,\partial_x \\
&\quad + (\cos\alpha\cos\gamma - \sin\alpha\cos\beta\sin\gamma)\,\partial_y + \sin\alpha\sin\beta\,\partial_z, \\
\mathcal{A}_3\big|_g &= \sin\beta\cos\gamma\,\partial_x + \sin\beta\sin\gamma\,\partial_y + \cos\beta\,\partial_z, \\
\mathcal{A}_4\big|_g &= \cos\alpha\cot\beta\,\partial_\alpha + \sin\alpha\,\partial_\beta - \frac{\cos\alpha}{\sin\beta}\,\partial_\gamma \quad \text{for } \beta \neq 0 \text{ and } \beta \neq \pi, \\
\mathcal{A}_5\big|_g &= -\sin\alpha\cot\beta\,\partial_\alpha + \cos\alpha\,\partial_\beta + \frac{\sin\alpha}{\sin\beta}\,\partial_\gamma \quad \text{for } \beta \neq 0 \text{ and } \beta \neq \pi, \\
\mathcal{A}_6\big|_g &= \partial_\alpha.
\end{aligned} \tag{7.64}$$

Here, $\mathcal{A}_3$ is tangent to the local orientation and plays the same role as ∂_ξ in $SE(2)$, $\mathcal{A}_1$ and $\mathcal{A}_2$ are orthogonal and comparable to ∂_η, and $\mathcal{A}_i$ with $i \in \{4,5,6\}$ are comparable to ∂_θ.

Expressed in the *second coordinate chart* cf. (7.8) the same left-invariant derivatives are given by

$$
\begin{aligned}
\mathcal{A}_1\big|_g &= \cos\check{\alpha}\cos\check{\beta}\,\partial_x + (\cos\check{\gamma}\sin\check{\alpha} + \cos\check{\alpha}\sin\check{\beta}\sin\check{\gamma})\,\partial_y \\
&\quad + (\sin\check{\alpha}\sin\check{\gamma} - \cos\check{\alpha}\cos\check{\gamma}\sin\check{\beta})\,\partial_z, \\
\mathcal{A}_2\big|_g &= -\sin\check{\alpha}\cos\check{\beta}\,\partial_x + (\cos\check{\alpha}\cos\check{\gamma} - \sin\check{\alpha}\sin\check{\beta}\sin\check{\gamma})\,\partial_y \\
&\quad + (\sin\check{\alpha}\sin\check{\beta}\cos\check{\gamma} + \cos\check{\alpha}\sin\check{\gamma})\,\partial_z, \\
\mathcal{A}_3\big|_g &= \sin\check{\beta}\,\partial_x - \cos\check{\beta}\sin\check{\gamma}\,\partial_y + \cos\check{\beta}\cos\check{\gamma}\,\partial_z, \\
\mathcal{A}_4\big|_g &= -\cos\check{\alpha}\tan\check{\beta}\,\partial_{\check{\alpha}} + \sin\check{\alpha}\,\partial_{\check{\beta}} + \frac{\cos\check{\alpha}}{\cos\check{\beta}}\,\partial_{\check{\gamma}} \quad \text{for } \check{\beta} \neq \frac{\pi}{2} \text{ and } \check{\beta} \neq -\frac{\pi}{2}, \\
\mathcal{A}_5\big|_g &= -\sin\check{\alpha}\tan\check{\beta}\,\partial_{\check{\alpha}} + \cos\check{\alpha}\,\partial_{\check{\beta}} - \frac{\sin\check{\alpha}}{\cos\check{\beta}}\,\partial_{\check{\gamma}} \quad \text{for } \check{\beta} \neq \frac{\pi}{2} \text{ and } \check{\beta} \neq -\frac{\pi}{2}, \\
\mathcal{A}_6\big|_g &= \partial_{\check{\alpha}},
\end{aligned}
\tag{7.65}
$$

with $\mathbf{R}_{(\alpha,\beta,\gamma)} = \mathbf{R}_{(\check{\alpha},\check{\beta},\check{\gamma})}$. The tangent space of $g \in SE(3)$ is spanned by these vector fields, i.e. $T_g(SE(3)) = \text{span}\{\mathcal{A}_1\big|_g, \mathcal{A}_2\big|_g, \mathcal{A}_3\big|_g, \mathcal{A}_4\big|_g, \mathcal{A}_5\big|_g, \mathcal{A}_6\big|_g\}$. The commutator relations can be obtained by complex trigonometric calculations $[\mathcal{A}_i, \mathcal{A}_j] = \mathcal{A}_i\mathcal{A}_j - \mathcal{A}_j\mathcal{A}_i$, which leads to exactly the same commutator relations as already defined in (7.39).

Notice that the left-invariant vectors at $g = e$ span the Lie algebra, i.e.

$$
\begin{aligned}
T_e(SE(3)) &= \text{span}\left\{\mathcal{A}_1\big|_e, \mathcal{A}_2\big|_e, \mathcal{A}_3\big|_e, \mathcal{A}_4\big|_e, \mathcal{A}_5\big|_e, \mathcal{A}_6\big|_e\right\} \\
&\text{with } \left\{\mathcal{A}_1\big|_e, \mathcal{A}_2\big|_e, \mathcal{A}_3\big|_e, \mathcal{A}_4\big|_e, \mathcal{A}_5\big|_e, \mathcal{A}_6\big|_e\right\} \\
&= \{\partial_x\big|_e, \partial_y\big|_e, \partial_z\big|_e, \partial_{\check{\gamma}}\big|_e, \partial_{\check{\beta}}\big|_e, \partial_{\check{\alpha}}\big|_e\}.
\end{aligned}
\tag{7.66}
$$

The left-invariant vectors $\mathcal{A}_j\big|_g$ form a basis for vector in the tangent space $T_g(SE(3))$, as explained in Subsections 2.8.1 and 2.8.3. A vector in the tangent space at $g \in SE(3)$ has the form $\sum_{j=1}^6 c^j \mathcal{A}_j\big|_g \in T_g(SE(3))$, with vector components $\boldsymbol{c} = (c^1, c^2, c^3, c^4, c^5, c^6)$ that have physical dimensions (length, length, length, 1, 1, 1). Similarly, a covector in a dual tangent space $T_g^*(SE(3))$ at $g \in SE(3)$ has the form $\sum_{j=1}^6 c_j d\mathcal{A}^j\big|_g \in T_g^*(SE(3))$ where $d\mathcal{A}^j$ are the dual vector fields of $\mathcal{A}_j$, with covector components $\hat{\boldsymbol{c}} = (c_1, c_2, \ldots, c_6)$ that have dimensions (1/length, 1/length, 1/length, 1, 1, 1). Furthermore, $\mathcal{A}_j$ also play the role of left-invariant differential operators on functions on $SE(3)$, as explained in

Subsection 2.8.2. They also establish the left-invariant gradient vector as

$$\nabla U(\boldsymbol{x},\mathbf{R}) = \begin{pmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \\ \mathcal{A}_4 \\ \mathcal{A}_5 \\ \mathcal{A}_6 \end{pmatrix} U(\boldsymbol{x},\mathbf{R}). \tag{7.67}$$

7.5.2 Left-invariant Derivatives and α-Right-Invariance

When (7.64) and (7.65) are used as left-invariant differential operators, we always have $\mathcal{A}_6\tilde{U}(g) = 0$ for all $g \in SE(3)$ where $\tilde{U}$ is α-right-invariant. This means that one only needs to consider the remaining five left-invariant derivatives $\mathcal{A}_i$ in case of α-right-invariant functions on $SE(3)$ and thus for functions on $\mathbb{R}^3 \rtimes S^2$. Note, however, that the left-invariant derivatives $\mathcal{A}_i\tilde{U}$ with $i \in \{1,2,4,5\}$, do not render α-right-invariant functions, since these left-invariant derivatives are dependent on the value of α resp. $\check{\alpha}$. When implementing operations on the domain $\mathbb{R}^3 \rtimes S^2$, for the calculations of left-invariant derivatives one can choose an arbitrary fixed value α_0 (e.g. $\alpha_0 = 0$) and use $\mathcal{A}_j\big|_{(\boldsymbol{x},\mathbf{R}_{(\alpha_0,\beta,\gamma)})}$. One should, however, always ensure that the result of the effective operation is independent on the choice of α_0.

To this end, we have the following important relation between the left-invariant derivatives at g_1 and g_2 iff $g_1 \sim g_2$ (that is, $g_1 = (\boldsymbol{x},\mathbf{R}_{(\alpha_1,\beta,\gamma)})$ and $g_2 = (\boldsymbol{x},\mathbf{R}_{(\alpha_2,\beta,\gamma)})$)

$$\nabla\tilde{U}(g_1) = \mathbf{Z}_{\alpha_1-\alpha_2}\nabla\tilde{U}(g_2), \quad \text{iff } g_1 \sim g_2, \tag{7.68}$$

where ∇ is defined in (7.67) and $\mathbf{Z}_{\alpha_1-\alpha_2}^{-1}$ "converts" the left-invariant gradient at g_2 to the left-invariant gradient at g_1 and is given by

$$\mathbf{Z}_\alpha = \left(\begin{array}{ccc|ccc} \cos\alpha & \sin\alpha & 0 & 0 & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & 0 & 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right). \tag{7.69}$$

In Subsection 7.7.3 this relation will be used to derive which diffusion equations are applicable on $\mathbb{R}^3 \rtimes S^2$.

7.5.3 Inner Product and Norm

Analogue to the inner product and norm described in Subsection 2.8.4 we define inner products and norms on the space of tangent vectors $T_g(SE(3))$ and covectors $T_g^*(SE(3))$.

The inner product is defined as

$$(\boldsymbol{c}, \boldsymbol{b})_\mu = \mu^2 (\sum_{j=1}^{3} c^j b^j) + \sum_{j=4}^{6} c^j b^j. \tag{7.70}$$

where μ can be interpreted exactly the same as μ in Subsection 2.8.4. The Grammian matrix is given by

$$\mathbf{G}_\mu = [(\mathcal{A}_i, \mathcal{A}_j)_\mu] = \operatorname{diag}(1, 1, 1, \mu^2, \mu^2, \mu^2), \tag{7.71}$$

and the Kronecker product is simply given by $\langle \hat{\boldsymbol{b}}, \boldsymbol{c} \rangle = \sum_{j=1}^{6} b_j c^j$. Consequently, the inner product between two covectors is given by

$$(\hat{\boldsymbol{c}}, \hat{\boldsymbol{b}})_\mu = \langle \hat{\boldsymbol{c}}, \mathbf{G}_\mu^{-1} \hat{\boldsymbol{b}} \rangle = \mu^{-2} (\sum_{j=1}^{3} c_j b_j) + \sum_{j=4}^{6} c_j b_j. \tag{7.72}$$

From the inner product on $T_g(SE(2))$ we can now induce a norm on vectors and covectors in the regular way, i.e. $\|\boldsymbol{c}\|_\mu = \sqrt{(\boldsymbol{c}, \boldsymbol{c})_\mu}$ and $\|\hat{\boldsymbol{c}}\|_\mu = \sqrt{(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}})_\mu}$.

7.5.4 Exponential Curves

The exponential curves of $SE(3)$ are found by

$$\gamma_{\boldsymbol{c}}(t) = \exp\left(t\, c^j \mathcal{A}_j\big|_e\right), \tag{7.73}$$

see Subsection 2.8.6 for an explanation of the concept of exponential curves. In matrix form they are found by

$$\mathbf{G}(t) = \exp\left(t c^j \mathbf{X}_j\right). \tag{7.74}$$

We aim to find the explicit expression for the spatial part of the exponential curves for $SE(3)$. First, we calculate the matrix form cf. (7.74). In the resulting 4×4 matrix $\mathbf{G}(t)$, the translation part is found in the fourth column, i.e. in the first

3 components of the vector $\mathbf{G}(t)\cdot(0,0,0,1)^{\mathrm{T}}$. This yields for $(c^4,c^5,c^6)\neq(0,0,0)$

$$\mathbb{P}_{\mathbb{R}^3}\gamma_{\mathbf{c}}(t)=\frac{1}{q^2}\begin{pmatrix} t\,c^4(C_{14}+C_{25}+C_{36})+(C_{26}-C_{35})(\cos(qt)-1)\\ \qquad\qquad+\dfrac{1}{q}\,(C_{166}+C_{155}-C_{364}-C_{245})\sin(qt)\\ t\,c^5(C_{14}+C_{25}+C_{36})+(C_{34}-C_{16})(\cos(qt)-1)\\ \qquad\qquad+\dfrac{1}{q}\,(C_{244}+C_{266}-C_{365}-C_{154})\sin(qt)\\ t\,c^6(C_{14}+C_{25}+C_{36})+(C_{15}-C_{24})(\cos(qt)-1)\\ \qquad\qquad+\dfrac{1}{q}\,(C_{344}+C_{355}-C_{265}-C_{164})\sin(qt)\end{pmatrix}, \tag{7.75}$$

where

$$C_{ij}=c^ic^j,\qquad C_{ijk}=c^ic^jc^k,\qquad q=\sqrt{C_{44}+C_{55}+C_{66}}, \tag{7.76}$$

and note that the super scripts c^i do not represent powers but indices of the vector components. The explicit expression for the rotation for $(c^4,c^5,c^6)\neq(0,0,0)$ is given by

$$\mathbb{P}_{SO(3)}\gamma_{\mathbf{c}}(t)=\\ \frac{1}{q^2}\begin{pmatrix} C_{44}+(C_{55}+C_{66})b & C_{45}(1-b)-q\,c^6a & C_{46}(1-b)+q\,c^5a\\ C_{45}(1-b)+q\,c^6a & C_{55}+(C_{44}+C_{66})b & C_{56}(1-b)-q\,c^4a\\ C_{46}(1-b)-q\,c^5a & C_{56}(1-b)+q\,c^4a & C_{66}+(C_{44}+C_{55})b\end{pmatrix}, \tag{7.77}$$

with $a=\sin(q\,t)$ and $b=\cos(q\,t)$.

For the case $c^4=c^5=c^6=0$ we obtain

$$\mathbb{P}_{\mathbb{R}^3}\gamma_{(c^1,c^2,c^3,0,0,0)}(t)=t\begin{pmatrix}c^1\\c^2\\c^3\end{pmatrix},\quad\text{and}\quad\mathbb{P}_{SO(3)}\gamma_{(c^1,c^2,c^3,0,0,0)}(t)=\mathbf{I}. \tag{7.78}$$

The exponential curves on $SE(3)$ simultaneously describe the exponential curves on $\mathbb{R}^3\rtimes S^2$, simply by a projection. That is, we can express the exponential curves explicitly using Euler angles, i.e. $\gamma_{\mathbf{c}}(t)=(\boldsymbol{x}(t),\alpha(t),\beta(t),\gamma(t))$ and subsequently dropping the α dimension, yielding $(\boldsymbol{x}(t),\beta(t),\gamma(t))$. Notice, however, that exponential curves on a group have the property that the tangent vector at $t=0$ uniquely describes the exponential curve. This property is lost for exponential curves on $\mathbb{R}^3\rtimes S^2$; the 5-dimensional tangent space is *not* uniquely describing the projected exponential curves. This fact becomes apparent in the next subsection where we will see that the torsion and curvature (properties that remain after projection on $\mathbb{R}^3\rtimes S^2$) are dependent on the entire 6-dimensional tangent vector $(c^1,c^2,c^3,c^4,c^5,c^6)$, which is element of the Lie algebra of $T_e(SE(3))$ and *not* of the tangent space $T_e(\mathbb{R}^3\rtimes S^2)$.

7.5.5 Curvature and Torsion

To obtain expressions for curvature and torsion we first calculate the spatial tangent vector, which yields

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbb{P}_{\mathbb{R}^3}\gamma_{\boldsymbol{c}}(t)) = \frac{1}{q^2}\begin{pmatrix} c^4(C_{14}+C_{25}+C_{36}) - q(C_{26}-C_{35})\sin(qt) \\ \qquad + (C_{166}+C_{155}-C_{346}-C_{245})\cos(qt) \\ c^5(C_{14}+C_{25}+C_{36}) - q(C_{34}-C_{16})\sin(qt) \\ \qquad + (C_{266}+C_{244}-C_{356}-C_{145})\cos(qt) \\ c^6(C_{14}+C_{25}+C_{36}) - q(C_{15}-C_{24})\sin(qt) \\ \qquad + (C_{355}+C_{344}-C_{256}-C_{146})\cos(qt) \end{pmatrix}, \tag{7.79}$$

where we assume $(c^1)^2 + (c^2)^2 + (c^3)^2 = 1$ to ensure that t is the arc length parametrization of the curve in $\mathbb{R}^3$. This can be easily enforced by rescaling $\boldsymbol{c}$ as $\boldsymbol{c} \leftarrow \frac{\boldsymbol{c}}{\sqrt{(c^1)^2+(c^2)^2+(c^3)^2}}$. We will use the assumption $(c^1)^2 + (c^2)^2 + (c^3)^2 = 1$ in all expressions in the rest of this subsection.

The curvature vector is now given by

$$\begin{aligned}\boldsymbol{\kappa}(t) &= \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbb{P}_{\mathbb{R}^3}\gamma_{\boldsymbol{c}}(t)) \\ &= \begin{pmatrix} (C_{35}-C_{26})\cos(qt) + \frac{1}{q}(C_{346}+C_{245}-C_{155}-C_{166})\sin(qt) \\ (C_{16}-C_{34})\cos(qt) + \frac{1}{q}(C_{355}+C_{145}-C_{244}-C_{266})\sin(qt) \\ (C_{24}-C_{15})\cos(qt) + \frac{1}{q}(C_{256}+C_{146}-C_{344}-C_{355})\sin(qt) \end{pmatrix}.\end{aligned} \tag{7.80}$$

From this equation we derive that the curvature norm for all values of t is given by

$$\begin{aligned}\kappa = \|\boldsymbol{\kappa}(t)\| = \big(&C_{11}(C_{55}+C_{66}) + C_{22}(C_{44}+C_{66}) + C_{33}(C_{44}+C_{55}) \\ &- 2C_{14}(C_{25}+C_{36}) - 2C_{2356}\big)^{\frac{1}{2}}.\end{aligned} \tag{7.81}$$

Using the Frenet formula the torsion vector is given by

$$\begin{aligned}\boldsymbol{\tau}(t) &= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{t}(t)\times\boldsymbol{\kappa}(t)) = \frac{C_{14}+C_{25}+C_{36}}{q} \\ &\times \begin{pmatrix} (C_{26}-C_{35})q\cos(qt) + (C_{155}+C_{166}-C_{245}-C_{346})\sin(qt) \\ (C_{34}-C_{16})q\cos(qt) + (C_{244}+C_{266}-C_{145}-C_{365})\sin(qt) \\ (C_{15}-C_{24})q\cos(qt) + (C_{344}+C_{355}-C_{146}-C_{256})\sin(qt) \end{pmatrix}.\end{aligned} \tag{7.82}$$

The torsion norm is therefore given by

$$\tau = \|\boldsymbol{\tau}(t)\| = \kappa\,\big|C_{36}+C_{25}+C_{14}\big|. \tag{7.83}$$

7.5.6 Deviation from Horizontality

In correspondence to Subsection 2.8.5, a smooth curve $q : \mathbb{R} \to SE(3)$, parameterized as $q(t) = (\boldsymbol{x}(t), \mathbf{R}(t))$ with $\|\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t)\| = 1$ is *horizontal* at position $t \in \mathbb{R}$ iff

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \mathbf{R}(t)\boldsymbol{e}_z, \tag{7.84}$$

i.e., the direction of $\mathbf{R}(t)\boldsymbol{e}_z$ should be tangent to the tangent vector of the spatial part of the curve. Notice that the definition of horizontality is independent of the α angle of $\mathbf{R}(t)$, so the definition is meaningful for exponential curves projected onto $\mathbb{R}^3 \rtimes S^2$ as well.

The *deviation from horizontality vector* $\boldsymbol{d}_{\mathrm{H}}$ of a curve at t is given by

$$\boldsymbol{d}_{\mathrm{H}}(t) = \mathbf{R}(t)^{-1}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t), \tag{7.85}$$

where $\boldsymbol{d}_{\mathrm{H}}(t) = \boldsymbol{e}_z$ means that the curve is horizontal at t.

The deviation from horizontality vector of exponential curves is constant over its parametrization and is given by

$$\boldsymbol{d}_{\mathrm{H}} = \left(\mathbb{P}_{SO(3)}\gamma_{\boldsymbol{c}}(t)\right)^{-1}\left(\mathbb{P}_{\mathbb{R}^3}\gamma_{\boldsymbol{c}}(t)\right) = \begin{pmatrix} c^1 \\ c^2 \\ c^3 \end{pmatrix}. \tag{7.86}$$

Exponential curves are horizontal if $(c^1, c^2, c^3) = (0, 0, 1)$.

7.6 Feature Estimation on 3D Orientation Scores

In this section we will describe, in analogy to Chapter 5, how one can estimate local features in 3D orientation scores. First, we will introduce the Hessian, and next, we will describe tangent vector estimation.

7.6.1 The Hessian

The general left-invariant Hessian on $SE(3)$ is defined by

$$\mathcal{H}U = \nabla(\nabla U) = \begin{pmatrix} \mathcal{A}_1^2 & \mathcal{A}_2\mathcal{A}_1 & \mathcal{A}_3\mathcal{A}_1 & \mathcal{A}_4\mathcal{A}_1 & \mathcal{A}_5\mathcal{A}_1 & \mathcal{A}_6\mathcal{A}_1 \\ \mathcal{A}_1\mathcal{A}_2 & \mathcal{A}_2^2 & \mathcal{A}_3\mathcal{A}_2 & \mathcal{A}_4\mathcal{A}_2 & \mathcal{A}_5\mathcal{A}_2 & \mathcal{A}_6\mathcal{A}_2 \\ \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_3^2 & \mathcal{A}_4\mathcal{A}_3 & \mathcal{A}_5\mathcal{A}_3 & \mathcal{A}_6\mathcal{A}_3 \\ \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_2\mathcal{A}_4 & \mathcal{A}_3\mathcal{A}_4 & \mathcal{A}_4^2 & \mathcal{A}_5\mathcal{A}_4 & \mathcal{A}_6\mathcal{A}_4 \\ \mathcal{A}_1\mathcal{A}_5 & \mathcal{A}_2\mathcal{A}_5 & \mathcal{A}_3\mathcal{A}_5 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5^2 & \mathcal{A}_6\mathcal{A}_5 \\ \mathcal{A}_1\mathcal{A}_6 & \mathcal{A}_2\mathcal{A}_6 & \mathcal{A}_3\mathcal{A}_6 & \mathcal{A}_4\mathcal{A}_6 & \mathcal{A}_5\mathcal{A}_6 & \mathcal{A}_6^2 \end{pmatrix} U. \tag{7.87}$$

On α-right-invariant functions we have the property $\mathcal{A}_6 U(g) = 0$ for all $g \in SE(3)$, and consequently [2] $\mathcal{A}_i \mathcal{A}_6 U = 0$ for all $i \in \{1,2,3,4,5,6\}$. Therefore, we first ensure the ordering $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6$, i.e. ensure that all derivatives are ordered such that angular derivative $\mathcal{A}_1$ always appears on the left-side and $\mathcal{A}_6$ always appears on the right-side, so angular derivatives are applied first, yielding

$$\mathcal{H}U = \begin{pmatrix} \mathcal{A}_1^2 & \mathcal{A}_1\mathcal{A}_2 & \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_1\mathcal{A}_5 - \mathcal{A}_3 & \mathcal{A}_1\mathcal{A}_6 + \mathcal{A}_2 \\ \mathcal{A}_1\mathcal{A}_2 & \mathcal{A}_2^2 & \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_4 + \mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_5 & \mathcal{A}_2\mathcal{A}_6 - \mathcal{A}_1 \\ \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_3^2 & \mathcal{A}_3\mathcal{A}_4 - \mathcal{A}_2 & \mathcal{A}_3\mathcal{A}_5 + \mathcal{A}_1 & \mathcal{A}_3\mathcal{A}_6 \\ \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_2\mathcal{A}_4 & \mathcal{A}_3\mathcal{A}_4 & \mathcal{A}_4^2 & \mathcal{A}_4\mathcal{A}_5 - \mathcal{A}_6 & \mathcal{A}_4\mathcal{A}_6 + \mathcal{A}_5 \\ \mathcal{A}_1\mathcal{A}_5 & \mathcal{A}_2\mathcal{A}_5 & \mathcal{A}_3\mathcal{A}_5 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5^2 & \mathcal{A}_5\mathcal{A}_6 - \mathcal{A}_4 \\ \mathcal{A}_1\mathcal{A}_6 & \mathcal{A}_2\mathcal{A}_6 & \mathcal{A}_3\mathcal{A}_6 & \mathcal{A}_4\mathcal{A}_6 & \mathcal{A}_5\mathcal{A}_6 & \mathcal{A}_6^2 \end{pmatrix} U. \tag{7.88}$$

Using $\mathcal{A}_6 \tilde{U} = 0$ we obtain the following expression for the Hessian on α-right-invariant functions $\tilde{U}$

$$\mathcal{H}\tilde{U} = \begin{pmatrix} \mathcal{A}_1^2 & \mathcal{A}_1\mathcal{A}_2 & \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_1\mathcal{A}_5 - \mathcal{A}_3 & \mathcal{A}_2 \\ \mathcal{A}_1\mathcal{A}_2 & \mathcal{A}_2^2 & \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_4 + \mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_5 & -\mathcal{A}_1 \\ \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_3^2 & \mathcal{A}_3\mathcal{A}_4 - \mathcal{A}_2 & \mathcal{A}_3\mathcal{A}_5 + \mathcal{A}_1 & 0 \\ \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_2\mathcal{A}_4 & \mathcal{A}_3\mathcal{A}_4 & \mathcal{A}_4^2 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5 \\ \mathcal{A}_1\mathcal{A}_5 & \mathcal{A}_2\mathcal{A}_5 & \mathcal{A}_3\mathcal{A}_5 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5^2 & -\mathcal{A}_4 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tilde{U}. \tag{7.89}$$

Since the sixth row is zero, one can consider this in fact is a 5×6 matrix.

7.6.2 Tangent Vector Estimation

Based on exactly the same reasoning as in Section 5.3, using the norm described in Subsection 7.5.3, we obtain for the optimum tangent vector $\boldsymbol{c}^*$

$$(\mathcal{H}U)^{\mathrm{T}} \mathbf{M}_\mu^2 (\mathcal{H}U) \boldsymbol{c}^* = \lambda \mathbf{M}_\mu^{-2} \boldsymbol{c}^*, \tag{7.90}$$

which can be rewritten as

$$(\mathbf{M}_\mu \mathcal{H}U \mathbf{M}_\mu)^{\mathrm{T}} (\mathbf{M}_\mu \mathcal{H}U \mathbf{M}_\mu)\, \tilde{\boldsymbol{c}}^* = \lambda\, \tilde{\boldsymbol{c}}^*, \tag{7.91}$$

where $\mathbf{M}_\mu = \mathrm{diag}(1/\mu, 1/\mu, 1/\mu, 1, 1, 1)$ and $\tilde{\boldsymbol{c}}^* = \mathbf{M}_\mu^{-1} \boldsymbol{c}^*$. This amounts to eigensystem analysis of the symmetric 6×6 matrix $(\mathbf{M}_\mu \mathcal{H}U \mathbf{M}_\mu)^{\mathrm{T}} (\mathbf{M}_\mu \mathcal{H}U \mathbf{M}_\mu)$, where one of the three eigenvectors gives $\tilde{\boldsymbol{c}}^*$. The eigenvector with the smallest corresponding eigenvalue is selected as tangent vector $\tilde{\boldsymbol{c}}^*$, and the desired tangent vector $\boldsymbol{c}^*$ is then given by $\boldsymbol{c}^* = \mathbf{M}_\mu \tilde{\boldsymbol{c}}^*$.

[2] Note that $\mathcal{A}_i \mathcal{A}_6 U = 0$ does not imply that $\mathcal{A}_6 \mathcal{A}_i U \neq 0$, but all $\mathcal{A}_6$ terms can be removed in that case using the commutator relations cf. (7.39)

By setting $c^1 = 0$ and $c^2 = 0$ we can enforce the deviation from horizontality to be zero. In the minimization term, $(\mathcal{H}U\boldsymbol{c}^*)$ can now be rewritten as

$$\begin{aligned}\mathcal{H}U\boldsymbol{c}\big|_{c^1=c^2=0} &= \mathcal{H}_{\text{hor}} W \boldsymbol{c}_{\text{hor}} \\ &= \begin{pmatrix} \mathcal{A}_1\mathcal{A}_3 & \mathcal{A}_1\mathcal{A}_4 & \mathcal{A}_1\mathcal{A}_5 - \mathcal{A}_3 & \mathcal{A}_2 \\ \mathcal{A}_2\mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_4 + \mathcal{A}_3 & \mathcal{A}_2\mathcal{A}_5 & -\mathcal{A}_1 \\ \mathcal{A}_3^2 & \mathcal{A}_3\mathcal{A}_4 - \mathcal{A}_2 & \mathcal{A}_3\mathcal{A}_5 + \mathcal{A}_1 & 0 \\ \mathcal{A}_3\mathcal{A}_4 & \mathcal{A}_4^2 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5 \\ \mathcal{A}_3\mathcal{A}_5 & \mathcal{A}_4\mathcal{A}_5 & \mathcal{A}_5^2 & -\mathcal{A}_4 \end{pmatrix} \begin{pmatrix} c^3 \\ c^4 \\ c^5 \\ c^6 \end{pmatrix}. \end{aligned} \tag{7.92}$$

Now the Euler Lagrange equation gives

$$(\mathbf{M}_\mu \, \mathcal{H}_{\text{hor}} U \, \mathbf{M}_{\mu,\text{hor}})^{\text{T}} (\mathbf{M}_\mu \, \mathcal{H}_{\text{hor}} U \, \mathbf{M}_{\mu,\text{hor}}) \, \tilde{\boldsymbol{c}}^*_{\text{hor}} = \lambda \, \tilde{\boldsymbol{c}}^*_{\text{hor}}, \tag{7.93}$$

where $\tilde{\boldsymbol{c}}^*_{\text{hor}} = \mathbf{M}_{\mu,\text{hor}} \boldsymbol{c}^*_{\text{hor}}$ and $\mathbf{M}_{\mu,\text{hor}} = \text{diag}\{1/\mu, 1/\mu, 1/\mu, 1\}$. This amounts to eigensystem analysis of a symmetric 4×4 matrix.

Once the local tangent vector is found, we can calculate the curvature, torsion, and deviation from horizontality as described in Subsection 7.5.5. Furthermore, analogously to Section 5.4, we define the measure for orientation confidence as

$$s = -\Delta_{\text{orth}} \tilde{U} = -\sum_{j=1}^{5} ((\boldsymbol{e}_j^{\text{orth}})^{\text{T}} \mathcal{H} U \boldsymbol{e}_j^{\text{orth}}, \tag{7.94}$$

where $\boldsymbol{e}_j^{\text{orth}}$ $(j \in \{1,2,3,4,5\})$ denote the 5 (eigen)vectors that are μ-orthogonal (i.e. $(\boldsymbol{e}_i^{\text{orth}}, \boldsymbol{e}_j^{\text{orth}})_\mu = \delta_{ij}$) to the estimated tangent vector.

7.7 Diffusion on 3D Orientation Scores

In this section we will first introduce diffusion on the sphere, which is introduced because it will be useful later on for implementation of diffusion on orientation scores. Subsequently we will introduce diffusion on $SE(3)$ and finally $\mathbb{R}^3 \rtimes S^2$.

7.7.1 Diffusion on the Sphere

The left invariant derivatives of $SO(3)$ are $\{\mathcal{A}_4, \mathcal{A}_5, \mathcal{A}_6\}$, cf. (7.64), so we can construct left-invariant diffusion processes using this set of derivatives. For functions on the sphere, we want to render diffusion operators without any dependence on α, i.e. α-right-invariant. The only diffusion operator fulfilling this requirement is the Laplace-Beltrami operator, which on $SO(3)$ is given by $\mathcal{A}_4^2 + \mathcal{A}_5^2 + \mathcal{A}_6^2$. By setting $\partial_\alpha = 0$ we obtain the Laplace-Beltrami operator on S^2,

$$\Delta_{S^2} = (\mathcal{A}_4^2 + \mathcal{A}_5^2 + \mathcal{A}_6^2)\Big|_{\partial_\alpha = 0} = \partial_\beta^2 + \frac{\cos\beta}{\sin\beta}\partial_\beta + \frac{1}{\sin^2\beta}\partial_\gamma^2. \tag{7.95}$$

The linear diffusion system on the sphere is therefore given by

$$\begin{cases} \partial_t a(\beta,\gamma,t) = \Delta_{S^2} a(\beta,\gamma,t), \\ a(\beta,\gamma,0) = f(\beta,\gamma), \end{cases} \tag{7.96}$$

where $a : S^2 \times \mathbb{R}^+$ represents the diffusion scale space of $f \in \mathbb{L}_2(S^2)$. The easiest way to apply this diffusion is via the spherical harmonics transform. The Laplace-Beltrami operator applied on spherical harmonics gives

$$\Delta_{S^2} Y_m^l = -l(l+1) Y_m^l, \tag{7.97}$$

so the spherical harmonic domain $\Delta_{S^2} f$ yields

$$\mathcal{F}_{S^2}[\Delta_{S^2} a(\cdot,\cdot,t)]_m^l = -l(l+1)\, \mathcal{F}_{S^2}[a(\cdot,\cdot,t)]_m^l. \tag{7.98}$$

Since the solution of (7.96) at time t is given by $e^{t\Delta_{S^2}} f$, the diffusion at time t in spherical harmonic domain is given by (see [85])

$$\mathcal{F}_{S^2}[e^{t\Delta_{S^2}} f]_m^l = e^{-t\, l(l+1)}\, \mathcal{F}_{S^2}[f]_m^l. \tag{7.99}$$

Alternatively, it can be expressed as an S^2-convolution

$$\begin{aligned} &(e^{t\Delta_{S^2}} f)(\beta,\gamma) = (\psi_{S^2}^t * {}_{S^2} f)(\beta,\gamma), \\ &\text{with} \quad \psi_{S^2}^t(\beta) = \sum_{l=0}^{\infty} e^{-tl(l+1)} \sqrt{\frac{2l+1}{4\pi}} P_m^l(\cos\beta), \end{aligned} \tag{7.100}$$

where the latter kernel is known as the Gauss-Weierstrass kernel [55].

7.7.2 Diffusion on $SE(3)$

The general left-invariant diffusion equation on $SE(3)$ is given by

$$\begin{cases} \partial_t W(g,t) = \nabla \cdot \mathbf{D} \nabla W(g,t) = \left(\sum_{i=1}^6 \sum_{j=1}^6 \mathcal{A}_j D_{ij} \mathcal{A}_i \right) W(g,t), \\ \partial_t W(g,0) = U(g), \end{cases} \tag{7.101}$$

where $W(\cdot,t)$ represents the diffused orientation score at time t.

In the 3D case, μ-isotropic diffusion cf. Subsection 2.9.1.1 becomes

$$\partial_t W = (\mathcal{A}_1^2 + \mathcal{A}_2^2 + \mathcal{A}_3^2 + \mu^2(\mathcal{A}_4^2 + \mathcal{A}_5^2 + \mathcal{A}_6^2))W. \tag{7.102}$$

Analogous to the special types of diffusion described in Subsection 2.9.1, we define diffusions suitable for enhancement of curves. Horizontal curvature- and torsion-free linear anisotropic diffusion along curves is defined as

$$\partial_t W = \left(D_{\mathrm{a}}(\mathcal{A}_1^2 + \mathcal{A}_2^2) + \mathcal{A}_3^2 + \mu^2 D_{\mathrm{a}}(\mathcal{A}_4^2 + \mathcal{A}_5^2 + \mathcal{A}_6^2) \right) W, \tag{7.103}$$

where D_{a} is the anisotropy factor. Finally, the diffusion along an exponential curve with tangent vector $\boldsymbol{c}$ is defined as

$$\mathbf{D}(\boldsymbol{c}, D_{\mathrm{a}}) = (1 - D_{\mathrm{a}}) \frac{\mu^2}{\|\boldsymbol{c}\|_\mu^2} \boldsymbol{c}\,\boldsymbol{c}^{\mathrm{T}} + D_{\mathrm{a}} \operatorname{diag}(1, 1, 1, \mu^2, \mu^2, \mu^2), \tag{7.104}$$

which allows to obtain curved kernels with torsion. The latter two types of diffusion are especially useful in the nonlinear (adaptive) case, where D_{a} and $\boldsymbol{c}$ are estimated using local features cf. Section 7.6. Notice that since both the diffusion equation and the used features are left-invariant, the resulting nonlinear diffusion equation is left-invariant as well.

In 3D, besides curves, another topic of interest is to enhance surfaces. The meaning of the spatial left-invariant vector fields changes; for a "horizontal surface", $\mathcal{A}_3$ is interpreted as the direction orthogonal to a surface, while $\mathcal{A}_1$ and $\mathcal{A}_2$ are tangents to a surface. Horizontal curvature- and torsion-free linear anisotropic diffusion along a surface is defined as

$$\partial_t W = \left((\mathcal{A}_1^2 + \mathcal{A}_2^2) + D_{\mathrm{a}}\mathcal{A}_3^2 + \mu^2 D_{\mathrm{a}}(\mathcal{A}_4^2 + \mathcal{A}_5^2 + \mathcal{A}_6^2)\right) W. \tag{7.105}$$

Finally, let $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ be two arbitrarily chosen orthogonal vectors that are both tangent to a surface in $\mathbb{R}^3$, then surface diffusion with inclusion of curvature and torsion is defined as

$$\begin{aligned} \mathbf{D}(\boldsymbol{c}, D_{\mathrm{a}}) =& (1 - D_{\mathrm{a}})\mu^2 \left(\frac{1}{\|\boldsymbol{c}_1\|_\mu^2} \boldsymbol{c}_1\,\boldsymbol{c}_1^{\mathrm{T}} + \frac{1}{\|\boldsymbol{c}_2\|_\mu^2} \boldsymbol{c}_2\,\boldsymbol{c}_2^{\mathrm{T}} \right) \\ &+ D_{\mathrm{a}} \operatorname{diag}(1, 1, 1, \mu^2, \mu^2, \mu^2). \end{aligned} \tag{7.106}$$

7.7.3 Diffusion on $\mathbb{R}^3 \rtimes S^2$

Diffusion on $\mathbb{R}^3 \rtimes S^2$ is considered as α-right-invariance-preserving diffusion on α-right-invariant functions on $SE(3)$. In this subsection we will derive which types of diffusions on $SE(3)$ cf. (7.101) preserve α-right-invariance, which will be called *valid* diffusions on $\mathbb{R}^3 \rtimes S^2$.

Remember equation (7.68), which shows the relation between left-invariant derivatives at two positions $g_1, g_2 \in SE(3)$ with $g_1 \sim g_2$. Inserting this equation in $\nabla \cdot \mathbf{D} \nabla \tilde{W}$ gives

$$\begin{aligned} \nabla \cdot \mathbf{D}(g_1) \nabla \tilde{W}(g_1) &= \nabla \cdot \mathbf{Z}_{\alpha_1 - \alpha_2}^{\mathrm{T}} \mathbf{D}(g_1) \mathbf{Z}_{\alpha_1 - \alpha_2} \nabla \tilde{W}(g_2) \\ &= \nabla \cdot \mathbf{D}(g_2) \nabla \tilde{W}(g_2), \quad \text{for all } g_1 \sim g_2, \end{aligned} \tag{7.107}$$

which shows that diffusion is only valid if

$$\mathbf{D}(g_1) = \mathbf{Z}_{\alpha_1 - \alpha_2} \mathbf{D}(g_2) \mathbf{Z}_{\alpha_1 - \alpha_2}^{\mathrm{T}}, \quad \text{for all } g_1 \sim g_2. \tag{7.108}$$

Next, we separately consider constant diffusions and adaptive diffusions.

7.7.3.1 Linear and Constant Diffusions

As already suggested by the requirement on convolution kernels cf. (7.61), the number of valid linear operations is limited. Suppose we have a diffusion tensor $\mathbf{D}$ that we use to apply a diffusion process to an α-right-invariant orientation score $\tilde{U}$. If $\mathbf{D}$ is an arbitrary diffusion tensor, which is not necessarily valid, one can always make it valid by taking the α-marginal to remove the dependency on α, i.e.

$$\begin{aligned}\partial_t \tilde{W}(g,t) &= \int_0^{2\pi} \nabla \cdot \mathbf{D} \nabla \tilde{W}(g) \mathrm{d}\alpha = \int_0^{2\pi} \nabla \cdot \mathbf{Z}_{\alpha-\alpha_0}^{\mathrm{T}} \mathbf{D} \mathbf{Z}_{\alpha-\alpha_0} \nabla \tilde{W}(g_0,t) \mathrm{d}\alpha \\ &= \nabla \cdot \left(\int_0^{2\pi} \mathbf{Z}_{\alpha-\alpha_0}^{\mathrm{T}} \mathbf{D} \mathbf{Z}_{\alpha-\alpha_0} \mathrm{d}\alpha \right) \nabla \tilde{W}(g_0,t) = \nabla \cdot \tilde{\mathbf{D}} \nabla \tilde{W}(g_0,t),\end{aligned} \tag{7.109}$$

where $g = (\boldsymbol{x}, \mathbf{R}_{(\alpha,\beta,\gamma)})$ and $g_0 = (\boldsymbol{x}, \mathbf{R}_{(\alpha_0,\beta,\gamma)})$. So the explicit calculation of the α-marginal can be omitted by considering only diffusion tensors $\tilde{\mathbf{D}}$ that can be constructed from arbitrary diffusion tensors $\mathbf{D}$ by

$$\tilde{\mathbf{D}} = \int_0^{2\pi} \mathbf{Z}_{\alpha-\alpha_0}^{\mathrm{T}} \mathbf{D} \mathbf{Z}_{\alpha-\alpha_0} \mathrm{d}\alpha = \int_0^{2\pi} \mathbf{Z}_{\alpha}^{\mathrm{T}} \mathbf{D} \mathbf{Z}_{\alpha} \mathrm{d}\alpha. \tag{7.110}$$

Notice that $\tilde{\mathbf{D}}$ always fulfills (7.108). From this relation we can conclude that the only remaining valid diffusion tensors $\tilde{\mathbf{D}}$ have the form $\tilde{\mathbf{D}} = \mathrm{diag}(A, A, B, C, C, 0)$ (where the sixth value is irrelevant since $\mathcal{A}_6 \tilde{U} = 0$), so the general equation for the constant linear diffusion on $\mathbb{R}^3 \rtimes S^2$-functions can be explicitly written as

$$\partial_t \tilde{W}(\boldsymbol{x},\beta,\gamma;t) = \begin{pmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \\ \mathcal{A}_4 \\ \mathcal{A}_5 \end{pmatrix}^{\mathrm{T}} \cdot \begin{pmatrix} A & 0 & 0 & 0 & 0 \\ 0 & A & 0 & 0 & 0 \\ 0 & 0 & B & 0 & 0 \\ 0 & 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 & C \end{pmatrix} \begin{pmatrix} \mathcal{A}_1 \\ \mathcal{A}_2 \\ \mathcal{A}_3 \\ \mathcal{A}_4 \\ \mathcal{A}_5 \end{pmatrix} \tilde{W}(\boldsymbol{x},\beta,\gamma;t). \tag{7.111}$$

This coincides with horizontal curvature- and torsion-free diffusion cf. (7.105) with $A = D_{\mathrm{a}}$, $B = 1$ and $C = \mu^2 D_{\mathrm{a}}$.

7.7.3.2 Adaptive Diffusions

In case of adaptive diffusions, both linear and nonlinear, the diffusion of type (7.111) with adaptive A, B, and C is valid as well, since the derivation in (7.109) can also be applied on an adaptive $\mathbf{D}$. However, we have more possibilities for $\mathbb{R}^3 \rtimes S^2$-diffusion. We will show that the diffusion is valid if Hessian features as

described in Section 7.6 are used, since in that case the arbitrariness of α_0 falls out of the diffusion equation.

By construction, (7.68) maps to the Hessian (cf. Section 7.6) as

$$\mathcal{H}\tilde{W}(g_1) = \mathbf{Z}_{\alpha_1-\alpha_2}\mathcal{H}\tilde{W}(g_2)\,\mathbf{Z}_{\alpha_1-\alpha_2}^{\mathrm{T}}, \quad \text{iff } g_1 \sim g_2, \tag{7.112}$$

and consequently, if tangent vectors $g \mapsto \boldsymbol{c}(g)$ are constructed using the Hessian,

$$\boldsymbol{c}(g_1) = \mathbf{Z}_{\alpha_1-\alpha_2}\boldsymbol{c}(g_2), \quad \text{iff } g_1 \sim g_2. \tag{7.113}$$

If the diffusion tensor has the form $\mathbf{D}(g_1) = \boldsymbol{c}(g_1)\,\boldsymbol{c}(g_1)^{\mathrm{T}}$ then inserting (7.113) gives

$$\mathbf{D}(g_1) = (\mathbf{Z}_{\alpha_1-\alpha_2}\boldsymbol{c}(g_2))(\mathbf{Z}_{\alpha_1-\alpha_2}\boldsymbol{c}(g_2))^{\mathrm{T}} = \mathbf{Z}_{\alpha_1-\alpha_2}\boldsymbol{c}(g_2)\boldsymbol{c}(g_2)^{\mathrm{T}}\mathbf{Z}_{\alpha_1-\alpha_2}^{\mathrm{T}}, \tag{7.114}$$

which matches requirement (7.108), meaning that this is a valid type of diffusion. The same holds if $\boldsymbol{c}$ would be acquired using other left-invariant matrix-based approaches such as the structure tensor.

Furthermore, the sum of two valid diffusion tensors $\mathbf{D}_1+\mathbf{D}_2$ forms a valid diffusion tensor again since

$$\begin{aligned}\mathbf{D}_1(g_1) + \mathbf{D}_2(g_1) &= \mathbf{Z}_{\alpha_1-\alpha_2}\mathbf{D}_1(g_2)\mathbf{Z}_{\alpha_1-\alpha_2}^{\mathrm{T}} + \mathbf{Z}_{\alpha_1-\alpha_2}\mathbf{D}_2(g_2)\mathbf{Z}_{\alpha_1-\alpha_2}^{\mathrm{T}} \\ &= \mathbf{Z}_{\alpha_1-\alpha_2}(\mathbf{D}_1(g_2) + \mathbf{D}_2(g_2))\mathbf{Z}_{\alpha_1-\alpha_2}^{\mathrm{T}}.\end{aligned} \tag{7.115}$$

Therefore, in an adaptive setting the diffusion along exponential curves cf. (7.104) is valid as well.

Since $\mathcal{A}_6\tilde{W}$ is always zero, and also the sixth component of $\mathbf{D}\nabla\tilde{W}(g_2)$ has no dependency on α, all cases of adaptive diffusion on $\mathbb{R}^3 \rtimes S^2$ can be expressed using only 5×5 diffusion tensor matrices and gradients with 5 components.

7.8 Implementations for 3D Orientation Scores

In this section we propose basic implementation schemes for 3D orientation score algorithms, focussing on orientation scores of the type $\mathbb{R}^3 \rtimes S^2 \to \mathbb{C}$.

First, we consider sampling of 3D orientation scores. The spatial coordinates of a 3D orientation score can be directly identified with a 3D spatial (image) volume and in practice we will use a Cartesian sampling grid in the spatial dimensions. For simplicity, we will assume that we have isotropic spatial voxels, i.e. the three spatial dimensions are sampled isotropically. Spherical sampling is more difficult, because it is not possible to obtain an equidistant sampling grid on the sphere for an arbitrary number of orientations. This will be the topic of the first subsection.

Subsequently, we will describe a implementations for the spherical harmonic transform, the $SE(3)$-convolution, left-invariant derivatives, and numerical schemes for (non)linear diffusion.

7.8.1 Sampling S^2 using Platonic solids

A problem with sampling 3D orientation scores is to uniformly distribute a finite number of sample points over the unit sphere. We consider two criteria that describe the uniformity of a chosen sampling:

- The distance between neighboring sampling points should be as equal as possible;
- Equal area around each sample point, i.e. if we calculate the Voronoi tessellation of the points on the sphere, the areas of all regions should be as similar as possible.

A common approach for sampling the sphere is to take tesselations of platonic solids [154] [42] [120], i.e. tetrahedron, cube or hexahedron, octahedron, dodecahedron, and icosahedron, see Figure 7.5. The vertices of the Platonic solids are exactly uniform concerning both criteria, so they provide a suitable choice for sampling points. The problem, however, is that no platonic solids exist in $\mathbb{R}^3$ with more vertices than the dodecahedron, which has 20 vertices. Therefore, a common approach is to approximate a uniform sampling on the sphere with more sampling points by applying a regular triangulation of the dodecahedron or icosahedron, followed by projection on the unit sphere.

A regular tesselation of order o means that the faces of the platonic solid, i.e. triangles in case of the icosahedron and pentagons in case of the dodecahedron, are subdivided into equally-sized triangles, where $o = 0$ means no subdivision. The process is illustrated in Figure 7.6 for the icosahedron, where the resulting

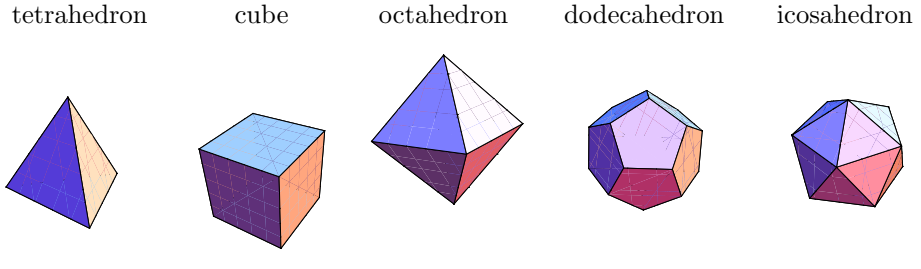


Figure 7.5: The platonic solids.

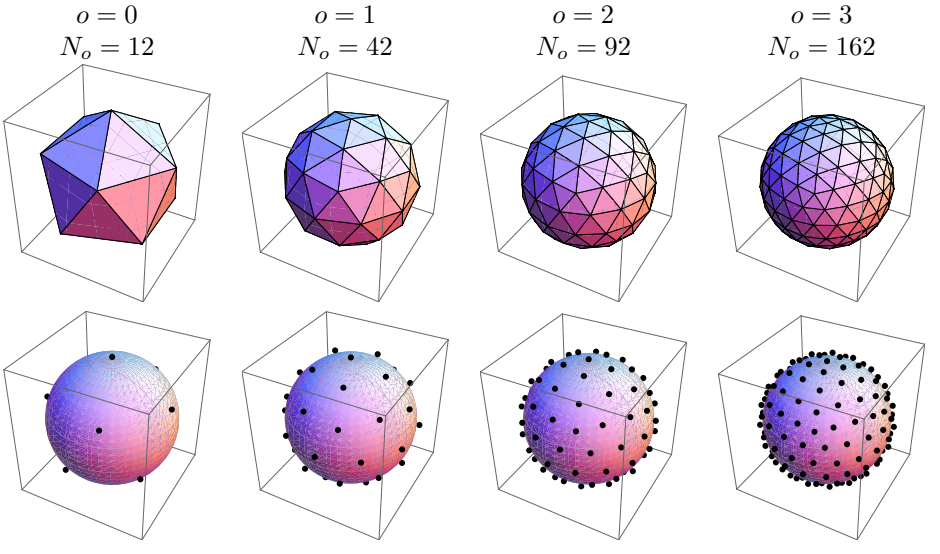


Figure 7.6: Equidistant sampling of the sphere using the 12 vertices of an icosahedron (left). To get more nearly-equidistant points, triangulations of order 1 or higher can be performed to divide each triangle in $(o+1)^2$ triangles, yielding $N_o = 2 + 10(o+1)^2$ vertices.

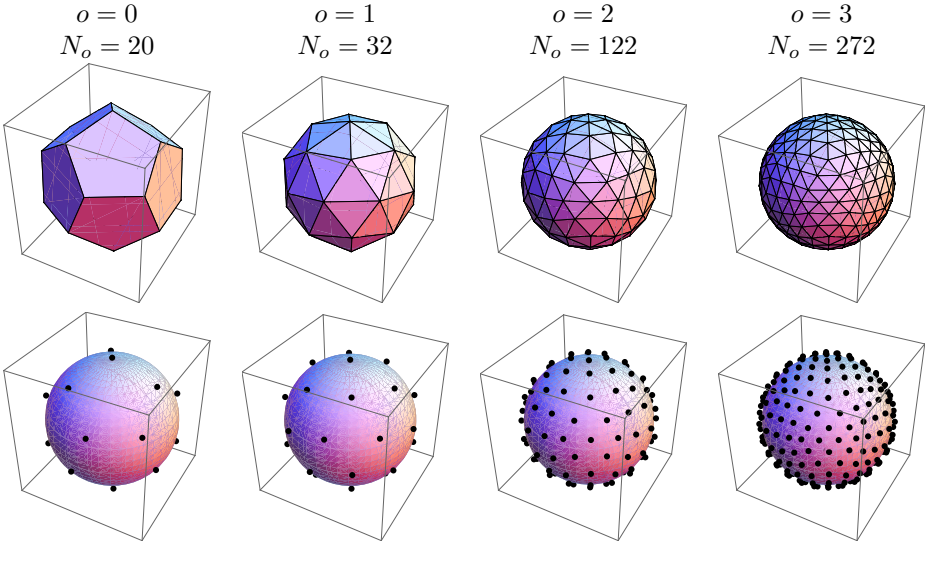


Figure 7.7: Equidistant sampling of the sphere using the 12 vertices of an dodecahedron (left). To get more nearly-equidistant points, a tessellation of order 1 divides each face into 5 triangles, leading to 32 vertices. Higher tessellation orders subdivide these triangles into smaller triangles yielding $N_o = 2 + 30(o+1)^2$ vertices.

number of sampling points is given by $N_o = 2 + 10(o+1)^2$. In case of the dodecahedron, a tesselation of order $o = 1$ divides every pentagon in 5 triangles; higher order tessellation subsequently divides these triangles into subtriangles, as illustrated in Figure 7.7. the resulting number of sampling points is given by $N_o = 20$ for $o = 0$ and $N_o = 2 + 30o^2$ for $o > 0$.

Similar to the 2D case, in many cases the 3D orientations score has a π-periodic symmetry, i.e. one does not always need to distinguish between a point on the sphere $\boldsymbol{n}$ and $-\boldsymbol{n}$ where $\boldsymbol{n} \in \{\mathbb{R}^3 \,|\, \|\boldsymbol{n}\| = 1\}$, In that case we want to obtain a uniform sampling of a half-sphere. Due to the symmetry of the platonic solids this can be obtained from the platonic solids by rejecting for instance all sampling points with $z < 0$.

7.8.1.1 Uniformity of a Spherical Sampling

In this subsection we aim to quantize the uniformity of a given spherical sampling. Assume we have a spherical sampling with a corresponding tessellation, i.e. all sampling points are identified with vertices and we have edges which connect to direct neighboring sampling points. Let $\mathbf{S} = \{\boldsymbol{v}_i | i \in \{1, 2, \ldots, N_o\}\}$ be the set of unit-length vectors describing the sampling points on the sphere. $\mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_i)$ is the set of the sampling points that are direct neighbors of $\boldsymbol{v}_i$, where the relation is reciprocal i.e. $\boldsymbol{v}_k \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_i) \leftrightarrow \boldsymbol{v}_i \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_k)$

The spherical distance of two points on a unit sphere is given by

$$d_{S^2}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \arccos\left(\frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \, \|\boldsymbol{v}_2\|}\right), \tag{7.116}$$

and the set of distances of all vertices to all of their direct neighbors is given by

$$\boldsymbol{d} = \left\{ d_{S^2}(\boldsymbol{v}_i, \boldsymbol{v}_j) \middle| i \in \{1, 2, \ldots, N_o\} \text{ and } \boldsymbol{v}_j \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_i) \right\}. \tag{7.117}$$

From these distances we calculate a measure for *distance uniformity*

$$u_{\text{dist}} = \frac{\max(\boldsymbol{d}) - \min(\boldsymbol{d})}{\text{mean}(\boldsymbol{d})}, \tag{7.118}$$

where 0 means exactly uniform.

The area of a spherical triangle on a unit sphere is given by

$$\begin{aligned} A(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3) = & d_{S^2}(\boldsymbol{t}_{\boldsymbol{v}_2}(\boldsymbol{v}_1), \boldsymbol{t}_{\boldsymbol{v}_3}(\boldsymbol{v}_1)) + d_{S^2}(\boldsymbol{t}_{\boldsymbol{v}_1}(\boldsymbol{v}_2), \boldsymbol{t}_{\boldsymbol{v}_3}(\boldsymbol{v}_2)) \\ & + d_{S^2}(\boldsymbol{t}_{\boldsymbol{v}_1}(\boldsymbol{v}_3), \boldsymbol{t}_{\boldsymbol{v}_2}(\boldsymbol{v}_3)) - \pi, \end{aligned} \tag{7.119}$$

where $\boldsymbol{t}_{\boldsymbol{p}}(\boldsymbol{v})$ gives the vector $\in \mathbb{R}^3$ that is orthogonal to $\boldsymbol{v}$ and that is situated in the plane spanned by $\text{span}\{\boldsymbol{v}, \boldsymbol{p}\}$, i.e.

$$\boldsymbol{t}_{\boldsymbol{p}}(\boldsymbol{v}) = \boldsymbol{p} - (\boldsymbol{v} \cdot \boldsymbol{p})\boldsymbol{v}, \tag{7.120}$$

	Icosahedron			Dodecahedron		
order	N_o	u_{dist}	u_{area}	N_o	u_{dist}	u_{area}
0	12	0	0	20	0	0
1	42	0.13	0.23	32	0.11	0.18
2	92	0.16	0.38	122	0.39	0.17
3	162	0.24	0.43	272	0.47	0.38
4	252	0.26	0.50	482	0.51	0.47
5	362	0.27	0.57	752	0.53	0.52

Table 7.1: Uniformity measures for the tessellations of icosahedron and dodecahedron. u_{dist} denotes the distance uniformity defined in (7.118) and u_{area} denotes the area uniformity defined in (7.123).

where we assume $\|\boldsymbol{p}\| = 1$ and $\|\boldsymbol{v}\| = 1$. We now define the set of neighboring triangles of a point $\boldsymbol{v}_i$ as

$$\mathcal{T}_{\mathbf{S}}(\boldsymbol{v}_i) = \{(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k) | \boldsymbol{v}_j \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_i) \text{ and } \boldsymbol{v}_k \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_i) \text{ and } \boldsymbol{v}_j \in \mathcal{N}_{\mathbf{S}}(\boldsymbol{v}_k)\}. \quad (7.121)$$

The effective area surrounding a sampling point $\boldsymbol{v}_i$ is now given by

$$A_{\mathbf{S}}(\boldsymbol{v}_i) = \frac{1}{3} \sum_{(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k) \in \mathcal{T}_{\mathbf{S}}(\boldsymbol{v}_i)} A(\boldsymbol{v}_i, \boldsymbol{v}_j, \boldsymbol{v}_k), \quad (7.122)$$

where we note that $\sum_{i=1}^{N_o} A_{\mathbf{S}}(\boldsymbol{v}_i) = 4\pi$, i.e. the sum of all areas matches the area of a total unit sphere. The measure for area uniformity is given by

$$u_{\text{area}} = \frac{\max_i(A_{\mathbf{S}}(\boldsymbol{v}_i)) - \min_i(A_{\mathbf{S}}(\boldsymbol{v}_i))}{\text{mean}_i(A_{\mathbf{S}}(\boldsymbol{v}_i))}, \quad (7.123)$$

where again 0 means exactly uniform.

Table 7.1 gives the uniformity measures for the tessellations of icosahedron and dodecahedron. We observe that the icosahedron tessellations perform better with respect to the distance uniformity, while the dodecahedron tessellations perform better with respect to the area uniformity. Depending on what criterion is most important for the algorithm in use, one should choose one of the two platonic solids as starting point for sampling on the sphere.

7.8.2 Implementation of the Spherical Harmonic Transform

We use a simple approach to calculate the spherical harmonic transform. In literature, several more efficient and accurate algorithms for discrete spherical harmonic transforms are proposed [31, 87, 12], which could be incorporated in future implementations.

When implementing the SHT of equation (7.23) we have to truncate l to the values $l \in \{0, 1, \ldots, L\}$, $L < \infty$. If we bandlimit $l \leq L$, the number of spherical harmonics is given by $n_{\mathrm{SH}} = (L+1)^2$. If we have a π-symmetric function on the sphere, we only need to consider the even values for l, i.e. $l \in \{0, 2, 4, \ldots, L\}$ where L is even. In that case the number of SH is given by $n_{\mathrm{SH}} = \frac{1}{2}(L+1)(L+2)$. It is hard to formulate a sampling theorem on the sphere for arbitrary samplings. However, from an algebraic point of view we can state that a spherical sampling with N_o points, the number of required spherical harmonics for reconstruction is lower bounded by $n_{\mathrm{SH}} \geq N_o$, which also gives a lower bound on L.

For the calculation of a discrete SHT we construct a matrix $\mathbf{M}$ of size $n_{\mathrm{SH}} \times N_o$ as follows

$$\mathbf{M} = [M_{jk}] = \frac{1}{\sqrt{C}} Y^{l(j)}_{m(j)}(\boldsymbol{v}_k), \quad j \in \{1, 2, \ldots, n_{\mathrm{SH}}\}, \; k \in \{1, 2, \ldots, N_o\}, \tag{7.124}$$

where j is a single index that enumerates the spherical harmonics, given by

$$\begin{aligned} \text{all values of } l\text{: } & j(l,m) = 1 + l^2 + (m+l), \\ \text{only even values of } l\text{: } & j(l,m) = 1 + \frac{l}{2}(l-1) + (m+l), \end{aligned} \tag{7.125}$$

and where $l(j)$ and $m(j)$ denote the inverse relations. Factor C in (7.124) is the normalization constant ensuring that $\sum_{j=1}^{n_{\mathrm{SH}}} \overline{M_{kj}} M_{jk} = 1$ for all k and is given by

$$C = \sum_{j=1}^{n_{\mathrm{SH}}} |Y^{l(j)}_{m(j)}(\beta, \gamma)| = \sum_{j=1}^{n_{\mathrm{SH}}} |Y^{l(j)}_{m(j)}(0, 0)|. \tag{7.126}$$

The discrete SHT on a sampled function on the sphere $\boldsymbol{f} = (f_1, f_2, \ldots, f_{N_o})$ is now given by

$$\boldsymbol{s} = (\mathbf{M} \cdot \mathrm{diag}(\boldsymbol{A}))\boldsymbol{f}, \quad \text{and inverse} \quad \boldsymbol{f} = (\mathrm{diag}(\boldsymbol{A}^{-1}) \cdot \overline{\mathbf{M}}^{\mathrm{T}})\boldsymbol{s}, \tag{7.127}$$

where $\boldsymbol{A}$ is a vector of weighting factors, given by

$$\boldsymbol{A} = (A_{\mathbf{S}}(\boldsymbol{v}_1), A_{\mathbf{S}}(\boldsymbol{v}_2), \ldots, A_{\mathbf{S}}(\boldsymbol{v}_{N_o})), \tag{7.128}$$

where $A_{\mathbf{S}}(\,)$is defined in (7.122). Vector $\boldsymbol{A}^{-1}$ denotes the componentwise inversion of the weighting factors in vector $\boldsymbol{A}$. These weighting factors compensate for differences in the surrounding areas of the sampling points, which is important since the spherical sampling is not exactly uniform, as has been discussed in Subsection 7.8.1.1. The effective operation of SHT followed by inverse SHT on $\boldsymbol{f}$ is given by $\mathbf{N} = (\mathbf{M} \cdot \mathrm{diag}(\boldsymbol{A})) \cdot (\mathrm{diag}(\boldsymbol{A}^{-1}) \cdot \overline{\mathbf{M}}^{\mathrm{T}}) = \overline{\mathbf{M}}^{\mathrm{T}} \cdot \mathbf{M}$ where $\mathbf{N}$ is the matrix of size $N_o \times N_o$ of the net operation. A perfectly invertible transformation would yield $\mathbf{N} = \mathbf{I}$. However in practice we only have $\mathbf{N} \approx \mathbf{I}$. Due to the construction the diagonal elements will all be exactly 1, but due to the truncation of the spherical harmonics there are no guarantees that the off-diagonal element

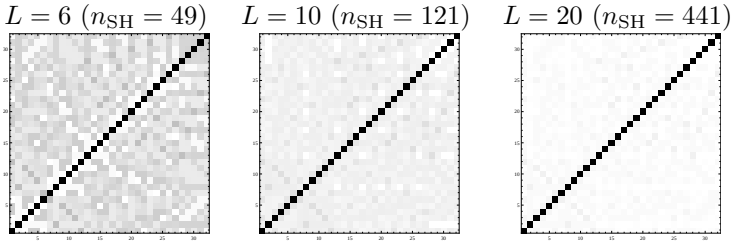


Figure 7.8: Effect of increasing spherical harmonic bandwidth L on the net operator matrix $\mathbf{N} = \overline{\mathbf{M}}^{\mathrm{T}}\mathbf{M}$ for the case $N_o = 32$ (first order tesselation of dodecahedron). Clearly, if one takes higher order spherical harmonics than strictly required, reconstruction of the sampled function on the sphere improves.

will all be exactly 0. The higher the value for L is chosen the more identical $\mathbf{N}$ becomes to the identity matrix, as illustrated in Figure 7.8. However, the price to pay when increasing L is a more redundant data representation and a potential decrease of the numerical stability; when one takes spherical derivatives via the SHT, the high order spherical harmonics will be strongly amplified.

Summarizing, due to the nonuniform sampling grid on the sphere, taking higher L than strictly necessary is sensible but L should not be made too large to prevent aliasing artefacts and instability.

7.8.3 Implementation of $SE(3)$-Convolutions

In this subsection we briefly discuss the implementation of the $SE(3)$ and $\mathbb{R}^3 \rtimes S^2$-convolution of Subsection 7.4.5. One could implement the convolution in a steerable way. Here we omit the derivation of steerable $SE(3)$-convolutions, as it is described extensively in [2, Chapter 6]. A similar approach is to use the $SE(3)$ fourier transform as described in [88, 22].

Alternatively, we can implement $SE(3)$-convolution in a non-steerable way cf. Subsection 3.3.3, which amounts to literally implementing the equation for $SE(3)$-convolution (7.52) or $\mathbb{R}^3 \rtimes S^2$-convolution (7.59). Such implementation is simpler and sometimes faster, depending on the size of the 3D orientation score and kernel, and the number of steerable components. In the 2D case, a set of sampled orientations over the circle exhibits the convenient property that they form a discrete subgroup of $SO(2)$. In 3D, however, we do not have this property, i.e. we cannot find an N-point sampling of $SO(3)$ such that all group products and group inverses map to another point in the sampling set. The same problem arises when sampling the sphere S^2. This problem can be handled in one of the following ways

- Often one has an analytical expression for ψ; in that case we can simply sample the kernel at any position whenever necessary.
- If we do not have an analytic solution, one can use some interpolation scheme, such as triangular interpolation using the 3 closest sampling points on the sphere, or using spherical harmonic interpolation. One also needs to interpolate the spatial coordinates, which can be done e.g. using spline interpolation.

In any case, $SE(3)$-convolutions have a very high computational complexity ($\mathcal{O}(NK)$ where N is the total number of sampled orientation score positions, and K the total number of sampled kernel positions. However, the operation is very suitable for parallel implementations, for instance on an FPGA. For optimal computational speed it is therefore of interest to look closely at the best order of loops, the way to store 3D orientation scores in memory to take advantage from caching, etcetera.

7.8.4 Implementation of Left-Invariant Derivatives

Spatial left-invariant derivatives $\mathcal{A}_1$, $\mathcal{A}_2$,and $\mathcal{A}_3$ can be implemented simply using finite differences. Using eq. (7.63), second order accurate centered finite differences are given by [3]

$$\begin{aligned}
\mathcal{A}_1 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x}+h\,\mathbf{R}\cdot\boldsymbol{e}_x,\mathbf{R})-U(\boldsymbol{x}-h\,\mathbf{R}\cdot\boldsymbol{e}_x,\mathbf{R})\right),\\
\mathcal{A}_2 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x}+h\,\mathbf{R}\cdot\boldsymbol{e}_y,\mathbf{R})-U(\boldsymbol{x}-h\,\mathbf{R}\cdot\boldsymbol{e}_y,\mathbf{R})\right),\\
\mathcal{A}_3 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x}+h\,\mathbf{R}\cdot\boldsymbol{e}_z,\mathbf{R})-U(\boldsymbol{x}-h\,\mathbf{R}\cdot\boldsymbol{e}_z,\mathbf{R})\right).
\end{aligned} \tag{7.129}$$

These finite differences can be calculated either by interpolation on the positions that are not on the sampling grid, or by first calculating finite differences corresponding to ∂_x, ∂_y, and ∂_z. The orientational derivatives are obtained similarly

$$\begin{aligned}
\mathcal{A}_4 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_h^{\boldsymbol{e}_x})-U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_{-h}^{\boldsymbol{e}_x})\right),\\
\mathcal{A}_5 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_h^{\boldsymbol{e}_y})-U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_{-h}^{\boldsymbol{e}_y})\right),\\
\mathcal{A}_6 U(\boldsymbol{x},\mathbf{R}) &\approx \frac{1}{2h}\left(U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_h^{\boldsymbol{e}_z})-U(\boldsymbol{x},\mathbf{R}\cdot\mathbf{R}_{-h}^{\boldsymbol{e}_z})\right),
\end{aligned} \tag{7.130}$$

where interpolations over $SO(3)$ need to be performed since the needed positions will most likely not correspond to sample positions.

[3] Note that we do not use the equation for left-invariant derivatives in either the first or second Euler angle coordinates chart cf. (7.64) and (7.65), since this would be a cumbersome detour. Instead we directly discretize the definition of left-invariant derivatives of equation (7.63).

In case of $\mathbb{R}^3 \rtimes S^2$ the spatial finite differences are calculated in the same way. The orientation derivatives $\mathcal{A}_4$ and $\mathcal{A}_5$ can be calculated as

$$\begin{aligned}
\mathcal{A}_4 u(\boldsymbol{x},\beta,\gamma) &\approx \frac{1}{2h}\left(u(\boldsymbol{x},\boldsymbol{\varpi}_{S^2}(\mathbf{R}_{(\alpha_0,\beta,\gamma)}\mathbf{R}_h^{\boldsymbol{e}_x})) - u(\boldsymbol{x},\boldsymbol{\varpi}_{S^2}(\mathbf{R}_{(\alpha_0,\beta,\gamma)}\mathbf{R}_{-h}^{\boldsymbol{e}_x}))\right),\\
\mathcal{A}_5 u(\boldsymbol{x},\beta,\gamma) &\approx \frac{1}{2h}\left(u(\boldsymbol{x},\boldsymbol{\varpi}_{S^2}(\mathbf{R}_{(\alpha_0,\beta,\gamma)}\mathbf{R}_h^{\boldsymbol{e}_y})) - u(\boldsymbol{x},\boldsymbol{\varpi}_{S^2}(\mathbf{R}_{(\alpha_0,\beta,\gamma)}\mathbf{R}_{-h}^{\boldsymbol{e}_y}))\right)\\
&\overset{\text{if } \alpha_0=0}{=} \frac{1}{2h}\left(u(\boldsymbol{x},\beta+h,\gamma) - u(\boldsymbol{x},\beta-h,\gamma)\right).
\end{aligned} \tag{7.131}$$

where $\boldsymbol{\varpi}_{S^2}$ takes the angles (β,γ) corresponding to the supplied rotation matrix. Angle α_0 can be arbitrarily chosen for each position $(\boldsymbol{x},\beta,\gamma) \in \mathbb{R}^3 \rtimes S^2$ separately. For the interpolation, the most straightforward methods are triangular interpolation and spherical harmonic interpolation. Triangular interpolation using the 3 closest sampling points on the sphere has the disadvantage that it leads to additional blurring, while spherical harmonic interpolation (that is, applying a SHT cf. (7.23) followed by calculation of response at the desired spherical position cf. (7.28)) can lead to overshoots and undershoots.

7.8.5 Numerical Schemes for Diffusion on 3D Orientation Scores

In this subsection we restrict ourselves to diffusions on $\mathbb{R}^3 \rtimes S^2$. First we will consider the linear case of (7.111). Subsequently, we will propose how to straightforwardly implement the nonlinear case.

7.8.5.1 Linear Diffusion

The linear diffusion equation on $\mathbb{R}^3 \rtimes S^2$ cf. (7.111) can be rewritten as (using $A = D_{\mathrm{a}}$, $B = 1$ and $C = \mu^2 D_{\mathrm{a}}$)

$$\partial_t \tilde{W}(\boldsymbol{x},\beta,\gamma,t) = \left(D_{\mathrm{a}}(\mathcal{A}_1+\mathcal{A}_2) + \mathcal{A}_3 + \mu^2 D_{\mathrm{a}}\,\Delta_{S^2}\right)\tilde{W}(\boldsymbol{x},\beta,\gamma,t). \tag{7.132}$$

To solve the equation numerically, the time derivative is a first order forward difference,

$$\partial_t \tilde{W}(\boldsymbol{x},\beta,\gamma,t) \approx \frac{\tilde{W}(\boldsymbol{x},\beta,\gamma,t+\tau) - \tilde{W}(\boldsymbol{x},\beta,\gamma,t)}{\tau}. \tag{7.133}$$

Spatially, we take second order centered finite differences for ∂_x^2, ∂_y^2, and ∂_z^2. In the orientation dimensions we use (7.98) to calculate Δ_{S^2} using the spherical harmonic transform, where for stability a small regularization with scale t_{reg} is applied via the spherical harmonic domain as well, i.e.

$$\mathcal{F}_{S^2}[\Delta_{S^2}\tilde{W}(\boldsymbol{x},\cdot,t)]_m^l = -l(l+1)\,e^{-t_{\mathrm{reg}}\,l(l+1)}\,\mathcal{F}_{S^2}[\tilde{W}(\boldsymbol{x},\cdot,t)]_m^l. \tag{7.134}$$

For efficiency, the chain of operators, SHT – (7.134) – inverse SHT, is stored in a $N_o \times N_o$ matrix, so that calculation of Δ_{S^2} at a single spatial position consists of a simple matrix-vector-multiplication.

Note that if one has the Green's function for diffusion on $SE(3)$, linear diffusion can be accomplished using an $SE(3)$-convolution with the Green's function rather than using the numerical scheme. It is questionable, however, whether this would be computationally more efficient.

7.8.5.2 Nonlinear Diffusion

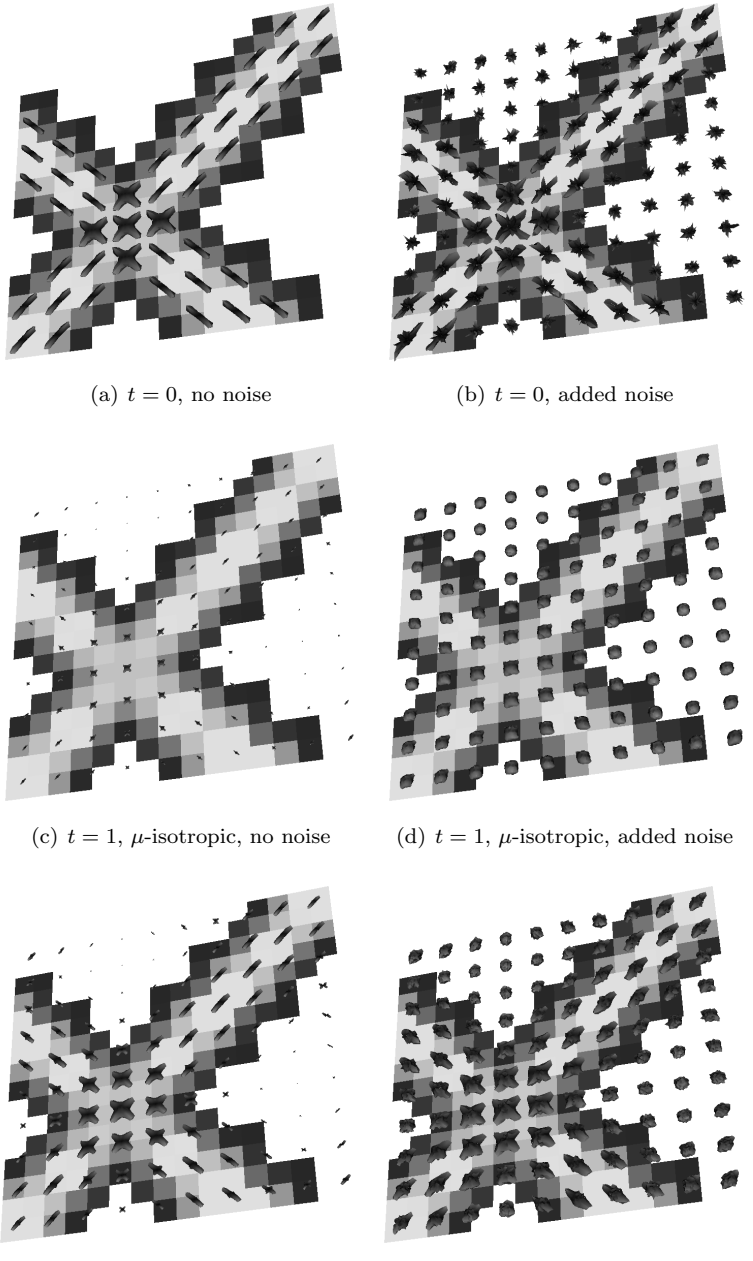
In the nonlinear case, we propose a numerical scheme similar to Subsection 6.4.1 for $SE(2)$-diffusion. The left-invariant derivatives are calculated as described in Subsection 7.8.4 and the time derivative is given by (7.133).

For nonlinear diffusion, we also need to estimate the local orientation confidence, and (optionally) curvature, torsion, and deviation from horizontality, see Subsection 7.7.2. For this purpose we need to calculate the Hessian cf. (7.89) using regularized derivatives. We propose to approximate these regularized derivatives by first applying regularization using the scheme for linear diffusion in Subsection 7.8.5.1, followed by taking finite differences as desrcribed in Subsection 7.8.4.

7.9 Results

Due to time restrictions and the difficulty of visualizing 3D orientation scores[4] we only implemented linear diffusion on 3D orientation scores. In Figures 7.9 and 7.10 we show preliminary results of the linear $SE(3)$-diffusion process. In these examples an artificial three-dimensional HARDI dataset is created, to which Rician noise is added. Next, we apply two different $SE(3)$-diffusions on both the noise-free and the noisy dataset. To visualize the result we use an experimental version of the DTI tool [32] which can visualize HARDI glyphs using the Q-ball visualization method [28]. In the results, all glyphs are scaled equivalently. The μ-isotropic diffusion does not preserve the anisotropy of the glyphs well; especially in the noisy case we observe that we get almost isotropic glyphs. With anisotropic diffusion, the anisotropy of the HARDI glyphs is preserved much better and in the noisy case the noise is clearly reduced. The resulting glyphs are, however, less directed than in the noise-free input image. This would improve when using nonlinear diffusion, or when adding some sort of "thinning" step in the method.

[4] Visualizing 3D orientation scores is a challenging problem and therefore the visualization of HARDI data is currently a PhD project of another PhD student within the BMIA group.



(a) $t = 0$, no noise

(b) $t = 0$, added noise

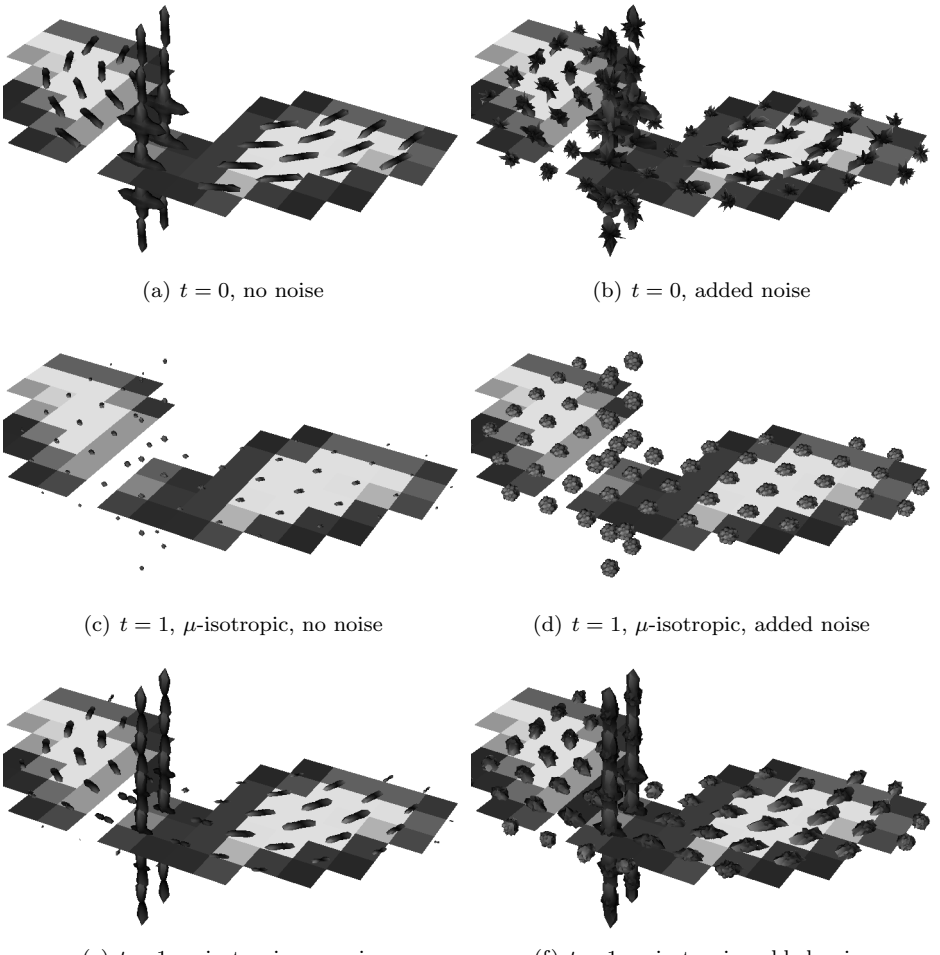
(c) $t = 1$, μ-isotropic, no noise

(d) $t = 1$, μ-isotropic, added noise

(e) $t = 1$, anisotropic, no noise

(f) $t = 1$, anisotropic, added noise

Figure 7.9: Result of $\mathbb{R}^3 \rtimes S^2$-diffusion on an artificial HARDI dataset of two crossing lines, with and without added Rician noise with $\sigma = 0.17$ (signal amplitude 1). Image size: $10 \times 10 \times 10$ spatial and 162 orientations. Parameters of the μ-isotropic diffusion process: $D_{\mathrm{a}} = 1$, $\mu = 0.1$, $t_{\mathrm{reg}} = .01$. Parameters of the anisotropic diffusion process: $D_{\mathrm{a}} = 0.01$, $\mu = 0.1$, $t_{\mathrm{reg}} = .01$.



(a) $t = 0$, no noise

(b) $t = 0$, added noise

(c) $t = 1$, μ-isotropic, no noise

(d) $t = 1$, μ-isotropic, added noise

(e) $t = 1$, anisotropic, no noise

(f) $t = 1$, anisotropic, added noise

Figure 7.10: Result of $\mathbb{R}^3 \rtimes S^2$-diffusion on an artificial HARDI dataset of two crossing lines where one of the lines is curved, with and without added Rician noise with $\sigma = 0.17$ (signal amplitude 1). Image size: $10 \times 10 \times 10$ spatial and 162 orientations. Parameters of the μ-isotropic diffusion process: $D_{\mathrm{a}} = 1$, $\mu = 0.1$, $t_{\mathrm{reg}} = .01$. Parameters of the anisotropic diffusion process: $D_{\mathrm{a}} = 0.01$, $\mu = 0.1$, $t_{\mathrm{reg}} = .01$.

7.10 Conclusions

In this chapter we have shown that we can map all techniques of the previous chapters on 2D orientation scores to the more complicated case of 3D orientation scores. Basically, everything stays the same, but some issues require more attention. Especially the fact that we usually have to deal with the coset space $SE(3)/(\mathbf{0} \times \mathrm{stab}(\boldsymbol{e}_z)) \cong \mathbb{R}^3 \rtimes S^2$ has been emphasized as an important issue. We have shown that we can consider functions $\mathbb{R}^3 \rtimes S^2 \to \mathbb{C}$ as functions on $SE(3)$ which are *α-right-invariant*, after which we were able to use all group theoretical results. The required preservation of α-right-invariance imposed additional constraints on e.g. the convolution kernel and the allowed types of (non)linear diffusion.

The implementations and results of this chapter are preliminary. However, the results suggest that even anisotropic linear diffusion on $SE(3)$ is a useful way to denoise HARDI data. Future work should include the implementation and evaluation of nonlinear $SE(3)$-diffusion; all recipes are included in this chapter. Furthermore, a more extensive evaluation should be performed to verify whether considering the entire $\mathbb{R}^3 \rtimes S^2$ as HARDI domain, instead of only the spatial or orientational domain as is usually done, is not only a conceptual but also a practical advantage.

Acknowledgements

Vesna Prckovska en Paulo Rodriguez are acknowledged for providing the artificial HARDI test images and the DTI tool supporting HARDI glyphs.

C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do, it blows away your whole leg.

Bjarne Stroustrup (1950)

8

MathVisionC++: a Library for Mathematical Image Processing

8.1 Introduction

The high-dimensional image processing algorithms described in this thesis pose severe requirements on computational power. This makes these algorithms challenging from an implementational point of view. This chapter therefore describes the design of MathVisionC++, a C++ library for mathematical image processing. First, we will describe the motivation for starting a new library. Then we will explain the design of the library and summarize the functionality it currently contains. The design patterns applied in the library are subsequently illustrated by an example: the implementation of generally applicable FIR filter algorithms.

Next, we describe our design of a convenient C++ interface to Mathlink, the communication protocol of Mathematica. This interface makes it convenient to make all MathVisionC++ functionality accessible from Mathematica. An important feature of the interface is the caching mechanism, which allows to reduce the number of large data transfers between C++ and Mathematica.

Note that this chapter is not meant to be a manual or a reference guide to the library. It only provides an introduction into the concepts.

8.1.1 Mathematica and MathVisionTools

The research described in this thesis has been performed within the Biomedical Image Analysis (BMIA) group at the Technische Universiteit Eindhoven. The member of this group frequently use Mathematica [153] for algorithm design, for both fundamental and applied medical image analysis research. This software has a number of useful properties for this purpose

- It is very convenient for fast translation of Mathematics (e.g. analytical formulas) to a numerical implementation (e.g. on pixel/voxel data);
- Implementing simple image processing algorithms is relatively fast, i.e. the so-called *rapid prototyping*;
- It provides useful built-in functionality for visualizing results, for instance **`ListDensityPlot`**, **`ListContourPlot`**, **`ListContourPlot3D`** etcetera;
- Mathematica is available for many platforms: Windows, Linux, MacOSX, etcetera.

These were the most important reasons to select Mathematica as standard prototyping environment for the research within BMIA.

However, no suitable add-ons were available to perform image analysis in Mathematica. Therefore, *MathVisionTools* [96] [2, Chapter 8] was developed. MathVisionTools is a powerful Mathematica package for image processing and analysis. The core of this library has been written in Mathematica. A lot of attention has been paid on optimizing the Mathematica code for speed, see [2, Section 8.2] for an example. Furthermore, the library is very flexible in the sense that many functions can be used on any-dimensional datasets and on both analytic functions and numeric data. The major features include

- Scale space and differential geometry, e.g. Gaussian derivatives for any order for and for any dimensions, and geometry driven diffusion;
- Orientation analysis, e.g. polar Fourier transform, 2D Hankel transform, $SE(2)$-convolution, and stochastic completion kernels;
- Commonly used image analysis functions, e.g. local maximum detection, B-spline interpolation, and region growing;
- Visualization functions, e.g. for visualizing tensorial images;
- Import / export functions, e.g. for the DICOM format, a widely-used file format for medical images.

Note that this is a non-exhaustive list, and new features are added continuously.

MathVisionTools has proved to be a useful library for the research within the BMIA group. However, firstly, algorithms developed in the group tend to grow more and more sophisticated, and require processing of higher-dimensional datasets such as orientation scores and scale spaces. Secondly, more and more algorithms developed in the group are becoming applicable to true medical imaging datasets, which are usually large datasets. These two developments result in an increased demand for processing large datasets. The disadvantages of Mathematica that start to hinder progress in some of the research are

- Mathematica is slow, since it is an interpreted language.
- Mathematica has a lot of redundant memory consumption: e.g. one cannot specify data types and no *pass-by-reference* construction exists.

The use of powerful Mathematica kernel servers to speed up calculations partly solves these problems, however this is not sufficient for many advanced algorithms.

8.1.2 Motivation for the Development of MathVisionC++

Because of the drawbacks of Mathematica and MathVisionTools described above, we started the development of *MathVisionC++*. The aim of MathVisionC++

was to develop a library with the following properties

1. *Efficiency*, concerning processing time but also concerning memory consumption;

2. Code *reusability*; the code should be as general as possible to prevent redundant coding of many largely overlap functions, in order to cover different cases. For instance, ideally a single image processing function in the library should be applicable to *any-dimensional* data with any numerical precision, without the need to change or add C++ code if the dimensionality or precision changes;

3. The library should be *platform independent*. Firstly since several different platforms are used within the group, and secondly since restricting to one platform would mean that the library is obsolete as soon as new platforms appear;

4. The library should contain built-in functionality for fundamental scale space and orientation score image processing research, but also for medical image analysis;

5. The functionality should be accessible from Mathematica, to be able to benefit from the speed and efficiency of MathVisionC++ and the advantages of Mathematica at the same time. However, it should also be possible to develop stand-alone programs entirely in C++.

8.1.3 Advantages of using C++

The programming language of our choice is *C++*. C++ is an open standard, so the language is by definition platform-independent, as long as no platform-dependent libraries are used. In addition, C++ is a compiler language which is very widely used, and therefore highly optimized compilers for several architectures exist (e.g. the Intel C++ Compiler).

We have chosen C++ rather than C for several reasons. Firstly, because a distinguishing feature of C++ compared to C are *templates*, which provide the perfect tool to achieve a high level of code reusability. For instance, in C one would need to write separate functions for convolution for different data types. In C++ one writes a single function where the data type and dimensionality are *template parameters*. The template parameters are resolved at *compile time*, meaning that the compiler *instantiates* a specific version of each function for all values of the template parameters that occur. Since the compiler knows the template parameters it can apply optimizations for each instantiation separately, to obtain optimally fast code.

A widespread misconception is that C++ is slower than C. However, C++ does not need to be slower, as long as one is a bit careful with some C++ features. The reputed slowness of C++ is mostly caused by a different programming style that arises from the object-oriented programming (OOP) paradigm. For instance in an "overdone" OOP design one could create a **class** `Image` that contains a method **virtual** `GetPixel` for random access to pixel data. This yields a flexible class design but a very slow access to image pixels. One should find a compromise between flexibility and speed. For instance, in this case it would be a better choice to declare the function **inline** `GetPixel`, which should lead to code that is just as fast as a low-level C-style get-pixel operations. Note, that this kind of constructions require more smartness from the C++ compiler, e.g. it should be able to inline the function in a proper way, but this should be no problem for most contemporary compilers.

Summarizing, for speed efficiency one should not aim at code reusability using compile time constructs such as **virtual** but instead use template programming constructions. Templates in C++ provide many possibilities. Veldhuizen [140] even shows that C++ templates are Turing complete by providing the recipe to implement any Turing machine with C++ templates. Programming using template constructions is called *template metaprogramming* [1] and it provides many "tricks" to increase speed at run-time. With the upcoming release of the C++0x standard, the new standard for C++ that is currently planned to be introduced in 2009, template programming will become even more powerful. Especially the introduction of so-called *concepts* it will be possible to create template metacode in a more structured way.

8.1.4 Other Image Processing Libraries

Many other image processing libraries exist, so one might wonder why we only considered MathVisionTools and MathVisionC++ till now. Other alternatives using high-level interpreted languages are

- The Digital Image Processing package for Mathematica [75]. This package does not provide the desired functionality: too slow and not sufficiently generic.
- Using Matlab with the Image Processing toolbox [97]. This environment basically has the same disadvantages as Mathematica: too slow and not generic.

There also exist numerous C/C++ libraries for image processing. The most interesting ones are

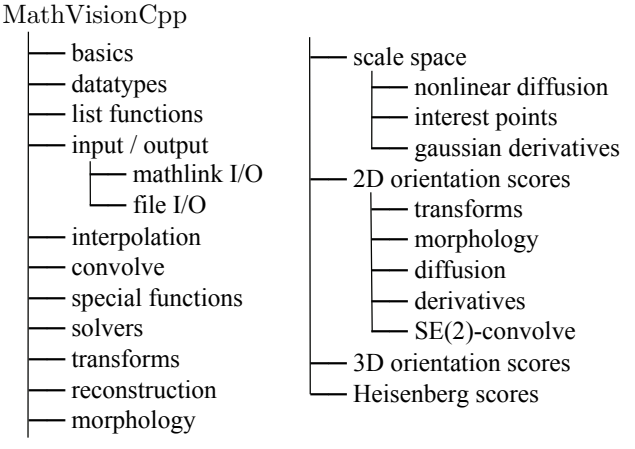


Figure 8.1: Structure of the MathVisionC++ library. Both the directory structure and the name spaces are organized in this way.

- C++ template image processing toolkit [131]. This library can handle 1-2-3-4D objects and works with C++ templates. It has a simple design with only four classes in the entire library. However, the flexibility is limited; all data structures are stored as 4D data structure `img` where all "unused" dimensions are set to size 1. This does not allow to optimize for lower dimensionalities. The entire library is implemented in a single header file, leading to slow compilation as the library grows. Furthermore, all algorithms are impemented in the `img` class itself, which is in our point of view not the right design, as will be discussed in the next section.

- Intel Integrated Performance Primitives [114]. This library is fast but it again lacks the required functionality, e.g. 2D only. Furthermore it is not freely available and it is platform dependent.

- Open CV [107]. This library is written with a different goal: real-time computer vision, so it does not contain the functions that we are interested in. Furthermore, it does not provide enough flexibility as it is 2D only.

- DIPlib [29] is an image processing library written in C, which includes a Matlab interface called DIPimage. It provides a lot of functionality, however it is written in C and therefore it is less flexible.

- ITK [122], this library is more specialized for segmentation and registration.

Summarizing, the major differences with our library are flexibility (any-D) and functionality, i.e. we need more mathematically-oriented image processing functions.

8.2 Library Functionality

In this section we provide a non-exhaustive list of image processing functions and other utility functions that are currently available. This is also summarized in Figure 8.1.

The library contains the following "any-dimensional" image processing functions

- FIR (Finite Impulse Response) filters, e.g. convolution, correlation, erosion, and dilation. See Section 8.4;
- Separable FIR filters;
- Inner-product taker;
- Various transforms: Fourier transform (i.e., wrappers to FFTW functions [57]), Zak transform and Gabor transform.
- Gaussian derivative and Gaussian derivative-at. Also including functions to calculate an entire Gaussian derivative jet efficiently.
- Spline interpolation using the algorithms described in [136].
- Explicit numerical schemes for nonlinear diffusion.

Furthermore, it contains the following "specific-dimensional" image processing functions

- 2D orientation scores (OS)
  - Orientation score transformation;
  - $SE(2)$-convolutions (steerable + non-steerable);
  - $SE(2)$-erosion and dilation;
  - OS Gaussian derivative jet;
  - OS nonlinear diffusion schemes;
- 3D orientation scores;
- Scale space interest points;
- Linear image reconstruction from a sparse set of features [78, 76];
- Heisenberg scores.

The utility functions, which are not directly related to image processing, include

- Functional programming constructs, such as `Map` to apply a given function to a sequence of elements;
- Multi-dimensional array processing, such as `Transpose` and `Take`;
- Linear algebra functions, such as `Dot` and `MatrixMultiply`;
- Convenient Mathlink C++ interface with a caching mechanism, see Section 8.5;
- `NumberTraits`, providing information on numeric types, see the example use in Subsection 8.4.2 for some more details;
- Several metafunctions to perform calculations at compile time;
- Parameter file parser to conveniently parse text files with e.g. algorithm parameters;
- Save / load PGM image file format;
- Save / load the MVA file format, which is the MathVisionC++ multi-dimensional array file format to store arrays of any dimension and any data type. This file format is also made accessible in Mathematica and ImageJ [115];
- Error classes for appropriate error handling.

8.3 Library Design

This section provides a global overview of the most important library design issues and the library structure. First we will list the most important design patterns, and subsequently describe some of these patterns in more detail.

The main design patterns are

- High-level data types such as multi-dimensional arrays are defined as C++ template classes (Subsection 8.3.1).
- Algorithms are implemented as *separate* template functions or template classes. I.e., algorithms are not member functions of data types, except for elementary functions such as `Dim` (the array dimensions), and `Sum` (the sum of all elements). This design pattern is described in more detail in Subsection 8.3.2.
- Names should better be long and clear than short and unclear.

- The library is organized in name spaces. All functions are put in namespace `MathVisionCpp`, which is divided into further categories, cf. Figure 8.1. Implementational details that are not relevant for the user of the library that need to reside in a header file, are put in a sub-**`namespace`** called `Detail`.
- Errors should be handled with the C++ **`try-catch-throw`** constructs.
- The library may depend on freely available *platform-independent* libraries only and should compile on all C++ compliant compilers.

8.3.1 Basic Data Types

High-level data types such as multi-dimensional arrays are implemented as C++ template classes. The most important data type implemented as template class is `MultiArray<T, Rank>`, which is a `Rank`-dimensional data type with elements of type `T`. For example

```
// 3D array of doubles :
MultiArray<double, 3> foo;
// 2D array of 2x2 float-matrices :
MultiArray<Matrix<float,2,2>, 2> bar;
```

`MultiArray` is comparable to a *packed array* in Mathematica, with the difference that Mathematica only provides packed arrays for the types **`int`**, **`double`**, `complex<`**`double`**`>`.

`MultiArray` provides all operators such as `*=`, `+=`, and sub-array iterators to iterate through the array. Furthermore, `MultiArray` provides simple memory management. An internal **`private`** variable **`bool`** `heapowner` specifies whether it "owns" the actual data its private pointer refers to. Only if `dataowner==`**`true`**, then the destructor of `MultiArray` frees the data using a customizable deallocator. This makes it convenient to let several `MultiArrays` refer to the same data.

Another data type is `Vect<T, Len>`, a `Len`-dimensional vector of type `T`. For example

```
Vect<int,3> myIntVec;       // 3-component integer vector
Vect<double,2> myDoubleVec; // 2-component double vector
```

Since `Rank` of `MultiArray` and `Len` of `Vect` are template parameters, the values are known at compile-time, which allows for more optimization by the compiler. Data type `Matrix<T,Rows,Cols>` is derived from `Vect` and can store matrices with compile-time known dimensions.

Other basic data types include `StridingMultiArray<T,Rank>`, a special version of `MultiArray<T,Rank>` where the data may be stored in memory with a certain *stride*, i.e. the number of locations in memory between successive array elements is flexible. This is useful to deal with sub-arrays of high-dimensional arrays without the need to *copy* this data to a new lower dimensional `MultiArray`, for example one can process a line in a 3D volume in either x, y, or z direction without the need to copy it. This is for instance useful for implementing *separable* convolutions by using an implementation of 1D convolutions.

8.3.2 Algorithms

Algorithms are not implemented as member functions of data types such as `MultiArray`. An algorithm is not an inherent property of a data object and therefore we argue that putting algorithms in data classes is bad practice.

Instead, in correspondence to the C++ STL (Standard Template Library) [129] algorithms are implemented as functions. The advantage is that a templated algorithm function often can be applied directly on different data types, without the need to duplicate any code for each data type it applies to. For example, algorithms on 1D data structures like `MultiArray<double,1>` should also work on STL data type `vector<double>` without the need to reimplement the algorithm. Since the basic data types of MathVisionC++ contain STL-compliant *iterators*, many of our algorithms can handle most STL data types and most STL algorithms also work with our data types.

More complicated algorithms, for instance ones that take many parameters or ones that require internal buffers, are implemented as classes. The advantage is that this the minimizes initialization time of the algorithm, as this only has to be done once for multiple calls. An example

```
{
  // constructor : algorithm setup
  GaussianDerivativeJet<T,Rank> MyGD(scale,
        maxDerivativeOrder, imageDimensions,
        boundaryConditions, convolutionMethod);
  // calculations
  MyGD.Calculate(result1, input1, 1);
  MyGD.Calculate(result2, input2, 1);
} // destructor : will now be called automatically
```

Note that object `MyGD` is constructed such that it is only suitable for a specific order of derivative jet, image dimensions, boundary conditions, and convolution method. In image processing chains it is quite common that multiple calculations with the same settings have to be performed. If one also needs other settings, one can either construct a second object for that case or, alternatively, most algorithm

classes contain a `Set` method to change the parameters of the algorithm object.

For convenience of the user of the library, we provide a function interface for most algorithms that are implemented as a classes. This provides a kind of shortcut to commonly used algorithms. For example to calculate a convolution we can write

```
ListConvolve(result, image, kernel, kernelCenter,
             boundaryCondition, convolutionMethod);
```

8.3.3 Functional Programming

We provide Mathematica-inspired functional programming constructs such as `Map`, for example

```
// declaration of variables
MultiArray<double,2> Image;  // ... load image
MultiArray<bool, 2> thresholdedImage;
double thresholdValue = ...; // some value

// apply threshold on Image using the Map_ function
Map_<0>(_1 > thresholdValue, thresholdedImage, Image);
```

where `Image` is the input, `thresholdedImage` is the output. The Boost lambda library [91] provides the possibility to write nameless functions objects like

```
pp{_1 > thresholdValue}
```

where "`_1`" acts as the first placeholder argument, similar to "**`#1`**" for pure functions in Mathematica. This function is applied on level `0` of the multidimensional array. Compare with Mathematica

```
thresholdedImage = Map[If[#1>thresholdvalue,1,0]&, Image, {2}];
```

Note: the *levelspec* convention in MathVisionC++ is chosen to be the other way around, i.e. in Mathematica the conversion would be

```
CppLevelSpec = Depth[data]-MathematicaLevelSpec-1
```

and in C++

```
MathematicaLevelSpec = data.ArrayRank-CppLevelSpec
```

where `ArrayRank` is a static data member of `data` yielding the dimensionality of `data`

We provide template functions `Map` and `Map_`. The difference is that `Map_` expects a single-argument function `f` that return its results, i.e. `result=f(input)`

while `Map` expects a function with two arguments where the result is passed-by-reference to the first argument, i.e. `f(result, input)`. The latter construction might be less esthetic but in many cases it is more efficient, since the assignment operator = requires to copy the return data. Some compilers, however, in some cases remove the copy redundancy. To gain speed, `Map` can also perform operations in parallel on computers with multiple cores or CPUs. In that case one should ensure that the function supplied to `Map` is thread-safe.

8.3.4 Dependencies on Other Libraries and Tools

The library currently depends on the following libraries

- Boost [91], a powerful extension to the C++ STL;
- FFTW, a fast library for Fourier transforms [57];
- Mathlink;
- BLAS/LAPACK, a library with fast basic linear algebra algorithms.
- UMFPACK, a library for solving sparse linear systems.

Furthermore we use

- Cmake, a cross-platform build environment [92];
- Doxygen, for documentation and tutorials [24];
- SVN, for version control.

8.4 Example Algorithm Design: FIR Filters

We aim to write a FIR (Finite Impulse Response) filter algorithm class that is as general as possible, to prevent rewriting a lot of similar code. The class should handle different data types and different dimensionalities. Furthermore, we want to be able to use the same code for convolution and correlation, as well as morphological operations. To make the code snippets below shorter, namespace scoops like "`DataTypes::`" will be omitted.

Convolution and correlation can be implemented both in the spatial domain ("direct") and in the Fourier domain ("Fourier"). In our design these two cases are implemented in separate classes `FIRFilterDirect` and `FIRFilterFourier`

since these algorithms contain totally different private data members. For example, a private data member of `FIRFilterFourier` is the `FourierCalculator`, which is not defined in `FIRFilterDirect`. Since it depends on the size of the kernel and the data which method is most efficient, we want to be able to select one of the two algorithm at run-time. For this purpose, we use class inheritance; we create an abstract base class `FIRFilterBase` which is defined as

```
// abstract base class
template <
   typename ImgType, int Rank,
   typename KrnType = ImgType,
   typename OutType =
      typename FIRFilterOutputTrait<ImgType,KrnType>::OutType >
class FIRFilterBase
{
  virtual void Calculate(MultiArray<OutType,Rank>& result,
              const MultiArray<ImgType,Rank>& img,
              const MultiArray<KrnType,Rank>& kernel,
              const Vect<int,Rank>& krnCent) = 0;

  virtual ~FIRFilterBase();
};
```

where the template parameters mean

- `ImgType` specifies the data type of the image, for instance **`int`**, **`double`**, or `complex<`**`double`**`>`.
- `Rank` specifies the dimensionality of the data, which is the same for both the input data, output data, and the kernel.
- `KrnType` specifies the data type of the kernel. By default this type is the same as the image type.
- `OutType` specifies the data type of the output array. By default it is determined automatically by `FIRFilterOutputTrait`, as will be explained in Subsection 8.4.2.2.

Subsequently, we derive both `FIRFilterDirect` and `FIRFilterFourier` from this base class, and provide implementations for the virtual function `Calculate`. The library contains the following declaration for `FIRFilterDirect`

```
template <
   class FilterMethod,
   typename ImgType, int Rank,
   typename KrnType = ImgType,
   typename OutType =
      typename FIRFilterOutputTrait<ImgType,KrnType>::OutType,
```

```
    class Algebra = PlusTimesAlgebra >
class FIRFilterDirect
       : public FIRFilterBase<ImgType,Rank,KrnType,OutType>
{
  /* ... implementation of constructor,
         Calculate, and destructor...   */
};
```

where the additional template parameters mean

- `FilterMethod` specifies whether correlation or convolution should be applied. The value should be either `Correlate` or `Convolve`.
- `Algebra` specifies the algebra to be used.

The constructor `FIRFilterDirect` is defined as

```
FIRFilterDirect(const Vect<int, Rank>& imdim,
                const Vect<int, Rank>& maxleft,
                const Vect<int, Rank>& maxright,
                BoundaryCondition bndcond);
```

The arguments of the constructor specify beforehand what the size of the image will be, and the maximum size of the kernel to the left and to the right measured from the kernel center (i.e. the kernel value at position $\mathbf{0}$). In this way the algorithm object can already initialize the calculation object to handle forthcoming calculations more efficiently, for instance allocating internal buffers, precalculating index maps for the appropriate boundary condition, etcetera. The declaration of class `FIRFilterFourier` and its constructor is similar, except that the template parameter `Algebra` falls out.

In both classes, the method `Calculate` adheres to the abstract declaration in `FIRFilterBase`. One can use pointers or references to `FIRFilterBase` to transparently work with `FIRFilterDirect` and `FIRFilterFourier`. Notice that a virtual function call to `Calculate` is slower, but a single virtual function call for an entire FIR filter operation does not lead to a significant decrease in speed.

8.4.1 Usage Examples

Suppose one wants to apply a convolution on a **double**-type 2D image with dimensions `img.Dim()` and with a **double**-type kernel `krn` with known dimensions 9×9 where the kernel center is exactly in the middle. One instantiates such algorithm object using

```
FIRFilterDirect<Convolve, double, 2> conv2D(
    img.Dim(), Vect<int,2>(4), Vect<int,2>(4), Cyclic);
```

The dimensions of the image and the kernel determine whether it is best to use the Fourier or direct implementation. A runtime decision whether to use a direct or Fourier implementation for convolution can be taken as follows

```
// determine what is the best method --> store in bestMethod
FIRFilterBase<double, 2>* conv2Dp;
if(bestMethod == ConvDirect)
  conv2Dp = new FIRFilterDirect<Convolve, double, 2>(...);
else // fourier
  conv2Dp = new FIRFilterFourier<Convolve, double, 2>(...);

// calculate a 2D convolution. At this point,
// we do not care which method is used to calculate this.
conv2Dp->Calculate(outArray, img, krn, Vect<int,2>(4));
```

If we want to use a complex-valued kernel, i.e. type `complex<double>`. We can use

```
FIRFilterDirect<Convolve, double, 2,
                complex<double> > conv2D(...);
```

Here, the `OutType` is determined automatically. One can overide this automatic determination by specifying another additional argument, e.g.

```
FIRFilterDirect<Convolve, float, 2, complex<float>,
                complex<double> > conv2D(...);
```

which ensures that the output has double precision while the input image and kernel have single precision.

Dilation and erosion on 2D images can be obtained by using the `MaxPlusAlgebra` resp. `MinMinusAlgebra`

```
FIRFilterDirect<Convolve, double, 2, double, double,
                MaxPlusAlgebra> dilate2D(...);
FIRFilterDirect<Correlate, double, 2, double, double,
                MinMinusAlgebra> erode2D(...);
```

where `Convolve` and `Correlate` are chosen such that the objects adhere to the most common definition of these operations.

It is possible to form illegal combinations of template parameters. For instance,

```
FIRFilterDirect<Convolve, complex<double>, 2, double,
                complex<double>, MaxPlusAlgebra> dilate2D(...);
```

is mathematically not defined. This will lead to compiler errors. For instance Microsoft Visual C++ 2008 Express renders the following error

```
error C2676: binary '<' : 'const std::complex<double>' does not define
             this operator or a conversion to a type acceptable to the
             predefined operator
```

This error makes clear that instantiating this type of FIR filter is conceptually wrong: no operator "<" is defined for complex numbers.

The implementation of the FIR filters can be applied in even more general cases. We can define a new data type `MyType`. If we have sensible definitions for multiplication and addition for `MyType`, we just need to implement the following two functions

```
inline MyType operator*(const MyType& t1, const MyType& t2);
inline MyType operator+(const MyType& t1, const MyType& t2);
```

Now we can simply instantiate a `FIRFilterDirect` on this type, i.e.

```
FIRFilterDirect<Convolve, MyType, 2> MyConvolver(...);
```

8.4.2 Implementational Details

Above, we only described the *interface* of the FIR filter classes. In this subsection we will provide more details on the programming paradigms that are used to implement the FIR filter classes such that they offer this optimal flexibility and efficiency.

8.4.2.1 Filter Method and Algebra

The first template parameter of `FIRFilterDirect` and `FIRFilterFourier` is `FilterMethod`. This must be either `Correlate` or `Convolve`. In fact, these two keywords represent **`struct`**s containing a number of simple **`static inline`** functions, for example

```
struct Correlate {
  template<typename T>
    static inline T& KernelIterIncrease(T &p)
    { return (++p); }
  // ... more static inline functions
};

struct Convolve {
  template<typename T>
    static inline T& KernelIterIncrease(T &p)
```

```
    { return (--p); }
  // ... more static inline functions
};
```

We see that both `Correlate` and `Convolve` contain a rather trivial function to "increase" an iterator or pointer to a kernel. For both correlation and convolution, the FIR filter function `Calculate` is the same piece of code, which consistently uses `FilterMethod::KernelIterIncrease(...)` to increase the iterator of the FIR filter kernel. In the `Convolve` case, however, increase actually amounts to a decrease, which ensures that the kernel is mirrored in the case of a convolution. `Correlate` and `Convolve` contain some more of these "trivial" functions, e.g. to initialize the kernel iterator, and to ensure the boundary conditions are handled in the appropriate way.

The specification of the algebra works in the same way. `PlusTimesAlgebra` is defined as

```
struct PlusTimesAlgebra
{
  template <typename T>
    static T Add(const T& arg1, const T& arg2)
    { return arg1+arg2; }

  template <typename Tout, typename Timg, typename Tkrn>
    static Tout Multiply(const Timg& img, const Tkrn& krn)
    { return img*krn; }

  // property : a == Add(IdentityValue(), a)
  template <typename T>
    static T IdentityValue()
    { return 0; }

  // property : a == Add(a, KernelIdentityValue())
  template <typename T>
    static T KernelIdentityValue()
    { return 0; }
};
```

For dilation `MinMinusAlgebra` is defined as

```
struct MinMinusAlgebra
{
  template <typename T>
    static T Add(const T& arg1, const T& arg2)
    { return std::min(arg1, arg2); }

  template <typename Tout, typename Timg, typename Tkrn>
    static Tout Multiply(const Timg& img, const Tkrn& krn)
```

```
    { return img-krn; }

  template <typename T>
    static T IdentityValue()
    { return std::numeric_limits<T>::max(); }

  template <typename T>
    static T KernelIdentityValue()
    { return -std::numeric_limits<T>::max(); }
};
```

where `IdentityValue` in theory should return an infinite value for this algebra, which is is "approximated" by `std::numeric_limits<T>::max()`, i.e. the highest value that data type `T` can contain. Algebra `MaxPlusAlgebra` is defined in analogous way.

8.4.2.2 Determining the Output Type

The next interesting question is how `FIRFilterOutputTrait` is defined, which is a template *metafunction* that "calculates" at compile-time what output type should come out. This is achieved as follows

```
// on default, IfThenElseType<Cond, T1, T2>::Type == T1
template <bool Condition, class T1, class T2>
struct IfThenElseType
{   typedef T1 Type; };

// template specialization for Cond == false, return type T2
template <class T1, class T2>
struct IfThenElseType<false, T1, T2>
{   typedef T2 Type; };

// heuristic automatic determination of OutType
template<typename ImgType, typename KrnType>
struct FIRFilterOutputTrait
{
  typedef typename IfThenElseType <
      /* if */    sizeof(KrnType)<sizeof(ImgType),
      /* then */  ImgType,
      /* else */  KrnType>
    ::Type OutType;
};
```

Above, `IfThenElseType` is a metafunction that returns either type `T1` or type `T2` depending on condtion `Cond`. Subsequently, `FIRFilterOutputTrait` uses the heuristic assumption that the output type `OutType` must be either equal

to ImgType or KrnType depending on which one is larger. For example, since **sizeof**(complex<**double**>) > **sizeof**(**double**), the output type should be complex<**double**>.

8.4.2.3 Fourier Spectrum Multiplication and Complex Numbers

Another interesting feature can be found in FIRFilterFourier. A real-valued dataset renders a *half* Fourier spectrum, since it is redundant to store a full spectrum in that case. If KrnType and ImgType are both real-valued or both complex-valued one can apply a multiplication of a half Fourier spectrum resp. full Fourier spectrum, which amounts to multiplying all elements of two MultiArrays. However, if KrnType and ImgType have different types, we need to multiply a full and a half spectrum. For this purpose, the library contains a algorithm object FourierSpectrumMultiplier that takes two boolean template values specifying whether the image resp. the kernel are real-valued, i.e.

```
template<bool ImgIsComplex, bool KrnIsComplex>
  struct FourierSpectrumMultiplier
  {
    template<class OutSpectrumType, class ImgSpectrumType,
             class KrnSpectrumType>
      static void Calculate(
          OutSpectrumType& OutSpectrum,
          const ImgSpectrumType& ImgSpectrum,
          const KrnSpectrumType& KrnSpectrum );
      {
        Multiply(OutSpectrum, ImgSpectrum, KrnSpectrum);
      }
  };
```

Next we need two specializations for the mixed cases.

```
template<>
  struct FourierSpectrumMultiplier<false, true>
  { /* specialized implementation of Calculate
       for full+half spectrum multiplication */
  };

template<>
  struct FourierSpectrumMultiplier<true, false>
  { /* specialized implementation ... */ };
```

The algorithm FIRFilterFourier contains the following code to multiply spectra.

```
FourierSpectrumMultiplier<
     NumberTraits<ImgType>::ComplexValued,
```

```
        NumberTraits<KrnType>::ComplexValued >
    :: Calculate(fourierProductBuf, fourierImgBuf, fourierKrnBuf);
```

where `NumberTraits<T>::ComplexValued` is a static boolean value that is true if type `T` is complex-valued. `NumberTraits` is a so-called *traits* class [129] that we implemented to obtain useful information on numeric types. For instance, the complex type corresponding to `T` can be obtained by `NumberTraits<T>::ComplexType`. This means that

```
NumberTraits<double>::ComplexType
  == NumberTraits<complex<double> >::ComplexType
  == complex<double>
```

This is useful to deduce the type of a Fourier transformed dataset. Analogously, `NumberTraits<T> ::RealType` gives the corresponding real-valued type. `NumberTraits<T>` also contains static functions `Conjugate`, `Real`, `Imag` which are not only applicable on complex-valued types (which is the case for the STL equivalent functions `conj`, `real`, and `imag`) but also have sensible definitions for real-valued types. This makes the coding of algorithms that work for both complex-valued and real-valued data much easier.

8.5 The Mathlink interface

We use Mathlink to make all MathVisionC++ functionality accessible from Mathematica. However, Mathlink has some disadvantages.

1. Data transfers are required from Mathlink-functions in C / C++ to the Mathematica kernel and vice versa, see Figure 8.2. This consumes a significant amount of time when working with large datasets
2. The provided Mathlink C interface is inconvenient to work with. The interface is too low-level and using it directly requires a lot of code duplication.

In this section we will describe our solutions to these problems. Our Mathlink caching mechanism is described in Subsection 8.5.1 and the C++ `MathLinkIO` class is described Subsection 8.5.2.

8.5.1 Caching Mechanism

To reduce Mathlink transfer overhead we designed a *caching-on-demand* system to keep data in MathVisionC++ instead of transferring it back. Figure 8.3 shows

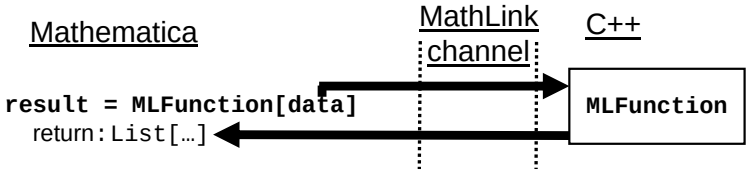


Figure 8.2: Illustration of calling a Mathlink function **`MLFunction`** from Mathematica. The data is transported over the Mathlink interface, either over a TCP/IP connection or using direct memory copying. The result has to be transported again. If **`data`** and / or the return data are large, this causes a lot of transfer overhead.

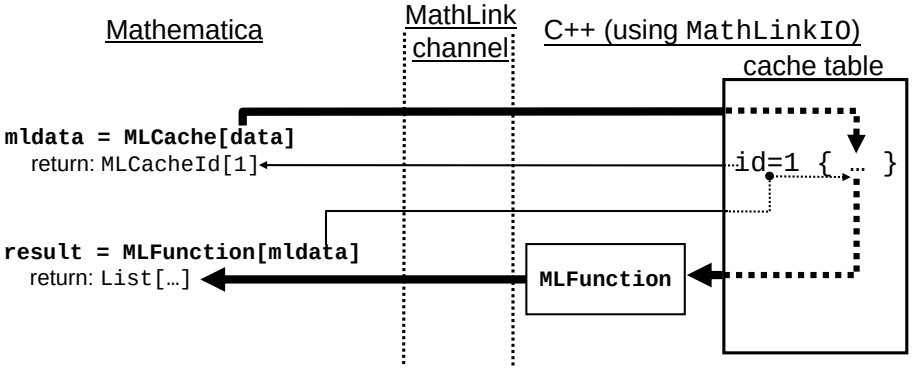


Figure 8.3: Example of caching data from Mathematica and using it in a Mathlink function call. See text (Subsection 8.5.1) for details.

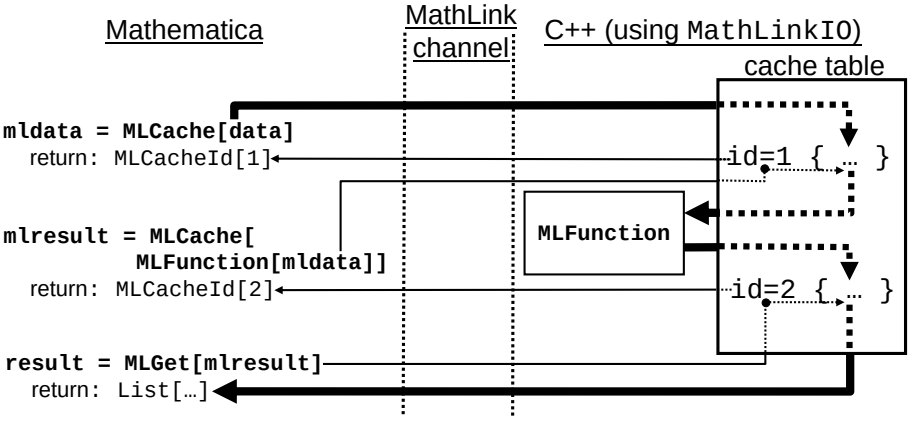


Figure 8.4: Example of caching the result of a Mathlink function instead of transferring it back to Mathematica. See text (Subsection 8.5.1) for details.

how data can be cached and used. The function **MLCache** transfers data to MathVisionC++, where the data is stored in a caching table. A unique identifier is assigned to the data, and is returned as **MLCacheId[***id***]**, which can be regarded as a *reference* to the data. The head **MLCachId** allows to recognize cached objects and deal with them transparently. That is, all **ML**-functions in MathVisionC++ can transparently be called with actual data as well as with cached data. This way of pre-caching data is useful if the same data is needed for multiple **ML**-function calls, i.e.

```
result1 = MLFunction1[mldata];
result2 = MLFunction2[mldata];
result3 = MLFunction3[mldata]; (* and so on ... *)
```

In the example of Figure 8.3, the return data still need the data to be transferred. However, Figure 8.4 shows that the **MLCache** can be used as well to wrap around an **ML**-function call. In this way, the result of **MLFunction** is stored in the cache table and an **MLCacheId** is returned. This is useful when **ML**-function calls are nested, i.e.

```
result1 = MLCache[MLFunction1[mldata]];
result2 = MLCache[MLFunction2[result1]];
result3 = MLCache[MLFunction3[result2]]; (* and so on ... *)
```

All arrays containing numerical data that Mathematica can store as *packed arrays* can currently be cached using this system.

MathVisionC++ includes some additional Mathematica functions for controlling the cache

- **MLRemove** and **MLRemoveAll** - to clear the cache;
- **MLCacheInfo** to obtain information on the stored elements;
- **MLReplace[id, expr]** to replace the data to which **id** (which should be an **MLCacheId**) points by **expr**. This is useful to prevent memory leaks in the cache when calling **MLCache** multiple times.

We will provide a usage example. First, we load the package and create an arbitrary random kernel and image.

```
In[1]:= <<MathVisionCpp`

In[2]:= img=Array[Random[]&,{16,16}];
        krn=Array[Random[]&,{5,6}];
```

We cache this data and both cached objects get their own unique identifier.

```
In[3]:= mlimg = MLCache[img]
```

```
Out[3]= MLCacheId[1]

 In[4]:= mlkrn = MLCache[krn]
Out[4]= MLCacheId[2]
```

Now we use these cached objects to do a list convolution using **MLListConvolve**, which is the MathVisionC++ equivalent to **ListConvolve**. However, suppose we make a typing mistake (underlined)

```
 In[5]:= mlresult=MLCache[MLListConvolve[mlkrn,mlomg,{3,3}]]
      MathVisionCpp::invalidparams: Invalid parameter types used. Function aborted.
Out[5]= MLInvalidArgumentNumber[2]
```

Notice that MathVisionC++ returns a clear error message specifying what goes wrong. **MLInvalidArgumentNumber[2]** specifies that the second argument is wrong, which makes debugging easier. This behavior is far more convenient for the user compared to the standard behavior of Mathematica, which proceeds the calculations symbolically without any error message.

We remove the typing mistake, yielding

```
 In[6]:= mlresult=MLCache[MLListConvolve[mlkrn,mlimg,{3,3}]]
Out[6]= MLCacheId[3]
```

If we now view the cache, we see that it contains these two datasets

```
 In[7]:= MLCacheInfo[]
Out[7]= {{MLCacheId[1],{16,16},{List,List},RealArray},
         {MLCacheId[2],{5,6},{List,List},RealArray},
         {MLCacheId[3],{16,16},{List,List},RealArray}}
```

Now we can transport the result back to Mathematica.

```
 In[8]:= result=MLGet[mlresult]; Dimensions[result]
Out[8]= {16,16}
```

When we want to reuse the variable **mlimg** and **mlresults** it is convenient to use **MLReplace**, which ensures no memory leak does occur.

```
 In[9]:= MLReplace[mlimg, Array[Random[]&,{100,80}]];
         MLReplace[mlresult,
                   MLListConvolve[mlkrn,mlimg,{3,3}]];
         MLCacheInfo[]
Out[9]= {{MLCacheId[1],{100,80},{List,List},RealArray},
         {MLCacheId[2],{5,6},{List,List},RealArray},
         {MLCacheId[4],{100,80},{List,List},RealArray}}
```

Note that **MLReplace** may also be called if the first argument is previously not used. In that case the behavior of **MLReplace[a,b]** is exactly similar to **a=MLCache[b]**.

The benefits of caching concerning speed are evaluated for some case studies in Section 8.6.

8.5.2 A Mathlink C++ class

To make writing the Mathlink interface functions in C++ less cumbersome compared to the standard C Mathlink interface, we designed **class** `MathLinkIO` as convenient C++ interface for Mathlink.

An instance of this class can be constructed to start communication via Mathlink. The default constructor will use `stdlink`, which is the standard Mathlink connection, but if desired a different Mathlink connection can be specified. The most important function in `MathlinkIO` are the `Get(...)` and `Return(...)` functions, that are overloaded for many different standard data types in order to easily transfer all these data types from and to Mathematica. These functions also transparently handle caching and data type conversion. If `MathlinkIO.Get(...)` obtains an **`MLCacheId[...]`** instead of true data it will obtain the data from the internal caching table instead. Furthermore, `MatlinkIO.Return(...)` will store the result in the cache rather than returning it, if an internal flag `cache_return` is set. This flag is automatically set and reset by the Mathematica functions **`MLCache`** and **`MLReplace`**, which are internally implemented differently for the usage cases of Figure 8.3 resp. Figure 8.4.

If a `Get` function obtains wrong argument types from Mathematica, for instance if it is supposed to store the result in a `MultiArray<double,2>` while the array supplied in Mathematica is three-dimensional, an `MLErrorType` object is **`throw`**n by `MathLinkIO`. The **`catch`** block sends an error message to Mathematica. The same **`try-catch-throw`** mechanism is used for any other Mathlink error that can occur.

Below we show an example in C++. `MLSquareImage` is a Mathlink function to square a two-dimensional array componenwise.

```
#include<mathvisioncpp/datatypes/multiarray.hpp>
#include<mathvisioncpp/inputoutput/mathlink_io/mathlink_io.hpp>
using namespace MathVisionCpp;

/* MPREP
  :Mathematica: MLSquareImage[image_]
*/
void MLSquareImage()
{
  IO::MathLinkIO mma;
  try {
    MultiArray<double,2> image;
    mma.Get(image, true); // true = in-place-processing
    image *= image;
    mma.Return(image);
  }
  catch (IO::MLErrorType& i)
```

```
    { mma.Abort(i); }
}
```

Note the `/*MPREP ... */` comment block. This is not just a comment, but an indicator for **mprepprep**, a small tool that we developed to automatically generate **mprep** files from the C++ code (see [101] for an explanation of **mprep** and MathLink). That is, in the example above the following **mprep** input file is created by **mprepprep**

```
:Begin:
:Function:       MLSquareImage
:Pattern:        MLSquareImage[image_]
:Arguments:      {image}
:ArgumentTypes:  {Manual}
:ReturnType:     Manual
:End:
```

8.6 Examples

All examples in this section were run on the **bigmath2** machine, which has the following specifications: 4 Dualcore Opteron 2.2Ghz CPUs, 64GB of RAM, CentOS Linux, Mathematica 6.02, and Gcc (GNU C/C++ compiler) version 4.1.2. The examples will demonstrate: Speed difference between MathVisionC++ and Mathematica / MathVisionTools, and benefits of using Mathlink caching.

8.6.1 Gradient Magnitude of a 3D Image

In Mathematica using MathVisionTools, calculating the gradient magnitude of an image can be done as follows

```
img = MLLoadArray["UltraSoundImage.mva"];
gradient = Transpose[GaussianDerivative[Sequence@@
   Transpose[{{20.,10.,16.}, RotateRight[{0,0,1},#]}],
   Method->Convolve][img]& /@ Range[3]],{4,1,2,3}];
gradmag = Map[Norm, gradient, {3}];
```

Here, **MLLoadArray** is an import function to load the MathVisionC++ multi-dimensional array file format, a file format that can store any `MultiArray` in an efficient way. The implementation in C++ using MathVisionC++ is

```
MultiArray<double, 3> img, gradmag;
MultiArray<double, 4> gradient;
LoadMultiArray("UltraSoundImage.mva", img);
```

	Direct	Fourier
Mathematica	13.6 s	34.1
MathVisionC++	2.9 s	3.3 s

Table 8.1: Timing results in seconds for gradient magnitude calculation on **bigmath2** using single-threaded code on an image with size $171 \times 100 \times 195$ and scales $t_x = 20$, $t_y = 10$, $t_z = 16$ and using the same kernel truncation criterion.

```
CalculateGaussianDerivativeJet(gradient,img,
   Vect3(20.,10.,16.),1,1,Cyclic,ConvDirect);
Map_<1>(Norm<2>(), gradmag, gradient);
```

Table 8.1 shows the timing results of these two implementations, for the case of convolution via the Fourier domain and direct convolution. Surprisingly, especially the Fourier method is much faster, which is probably because of the FFTW library, which is known to contain very efficient implementations of the FFT algorithm.

8.6.2 Gaussian Derivative-at

In this example, we show the advantage of Mathlink caching. We use the MathVisionC++ library from within Mathematica and compare the speed to MathVisionTools. "Gaussian derivative-at" is a function that calculates a single Gaussian derivative at a single scale space position (i.e. a position and a single scale). When estimating optic flow using top points [30] [77] a lot of Gaussian derivative-at calculations take place on the same 2D+time image. Table 8.2 shows timing results of the MathVisionTools implementation **GaussianDerivativeAt** versus the MathVisionC++ implementation **MLGaussianDerivativeAt**, showing that the MathVisionC++ implementation is much times faster than the MathVisionTools implementation. Furthermore, it shows that when not using Mathlink caching, a tremendous amount of time gets lost in transferring data.

For the optic flow case, another solution is to let **MLGaussianDerivativeAt** accept a list of scale space coordinates rather than only a single one. However, Gaussian derivative-at is also commonly used in iterative procedures e.g. to refine top points [112], so if one wants the main iteration to be programmed in Mathematica, caching is beneficial.

8.6.3 Interpolation

Many functions in MathVisionC++ are faster than built-in Mathematica functions. For example, in Mathematica, linear interpolation of a list of points on an

	total time (excl. transfer)				transfer
	Truncate	Cyclic	Reflective	Constant	time
MathVisionTools	0.43 s	0.14 s	0.24 s	0.24 s	-
MathVisionC++	0.0010 s	0.0018 s	0.0019 s	0.0017 s	0.16 s

Table 8.2: Timing results in seconds for Gaussian derivative-at on **bigmath2** using single-threaded code on a 3D dataset with dimensions $200 \times 200 \times 10$) for four commonly used boundary conditions. Timings are based on an average of 500 Gaussian derivatives with random scales between $t = 3$ and $t = 50$ and random derivative orders between 0 and 3, which is a realistic distribution of Gaussian derivative-at parameters when used in the optic flow method.

	Total calculation time (including transfer)	Total transfer time
Mathematica	8.8 s	-
MathVisionC++	2.5 s	0.8 s

Table 8.3: Timing results in seconds for linear interpolation on **bigmath2** using single-threaded code on a 512 x 512 image for 1000000 interpolations.

image **img** is achieved by

```
imgintMma = ListInterpolation[img, InterpolationOrder->1]
results = imgintMma@@@{{x1,y1},{x2,y2},...}
```

Using MathVisionC++ a new function is available to do the same

```
imgintCpp = MLNewBSplineInterpolator[mlimg,1]
results = MLBSplineInterpolatedValue[imgintCpp,
                                     {{x1,y1},{x2,y2},...}]
```

Table 8.3 shows the timing results. Clearly, MathVisionC++ is much faster, and despite the Mathlink transfer overhead, we can still take advantage of this speed in Mathematica.

8.7 Conclusions

We introduced MathVisionC++: a C++ library for image processing. The major difference with other C(++) libraries is that it provides much more flexibility concerning any-dimensionality of the data and more mathematically-oriented functions. In that sense, "image processing library" might not be covering the whole scoop of the library, a better term is "multi-dimensional signal processing library". Since we widely use template programming, the library is designed to be extremely generic and flexible, without paying the price for speed as is usually

paid for genericity, for instance in Mathematica. A lot of the C++ implementations prove to be faster than their Mathematica counterparts.

We provide a full interface to Mathematica, to take advantage of strengths of both Mathematica and C++. For this purpose we created a C++ interface to Mathlink that is far more convenient than the standard C interface. The included caching mechanism allows for a large Mathlink transfer overhead reduction. We emphasize, however, that the MathVisionC++ library is not only useful together with Mathematica. It is perfectly suitable to develop full-C++ image processing applications. Furthermore, we could easily provide interfaces to other programs, such as Matlab.

Because of the generic design patterns we expect that MathVisionC++ can provide a good basis for both future image processing research and advanced image processing applications.

Acknowledgements

The library described in this chapter has been developed together with Bart Janssen. Research assistent Roel Jordans is acknowledged for his substantial contributions to the code and documentation.

The important thing is not to stop questioning.

Albert Einstein (1879–1955)

9 Summary and Future Research

9.1 Enhancement of Crossing Elongated Structures in Images (Summary)

The enhancement of structures in noisy image data is relevant for many (bio)medical imaging applications. Existing generic image processing methods cannot appropriately handle elongated structures that *cross* each other. In this thesis we propose to solve this problem by using so-called *orientation scores* of images.

The basics of orientation scores are introduced in Chapter 2. A two-dimensional image is convolved with a rotating anisotropic filter kernel, resulting in an orientation score, which is a three-dimensional dataset where orientation is the third dimension. By ensuring that all rotated filters together cover all relevant frequencies, this transformation is made invertible.

Orientation scores can be considered as functions on the Euclidean motion group $SE(2)$. This observation makes it possible to map many well-known image processing algorithms to the orientation score domain $SE(2)$. In this thesis we construct operations on orientation scores, for the purpose of enhancing the corresponding image *via* its orientation score. We provide an explanation of the necessary Lie group theory that we subsequently need to derive a few useful theoretical results. Firstly, we show that if the effective operations on the image are required to be rotation invariant and translation invariant, which is usually desirable, the corresponding operation in the orientation score should be *left-invariant*. Therefore we ensure that all subsequent operations on orientation scores are left-invariant, and we start with the derivation of left-invariant partial derivatives. Secondly, we briefly introduce the differential geometry on $SE(2)$, which is important to describe local structures in the orientation score. We introduce the *exponential curves* in $SE(2)$, which are shown to be spirals in $SE(2)$ that are equivalent to straight tangent lines in $\mathbb{R}^n$. Thirdly, we introduce left-invariant diffusion equations and convection-diffusion equations in $SE(2)$.

Subsequently, in Chapter 3 we aim at linear operations on orientation scores by studying the $SE(2)$-convolution in more detail. We show that we can implement such convolutions using *steerable filters*, leading to more efficient implementations if the $SE(2)$-kernel is sufficiently angularly bandlimited. We also establish the relation between steerable $SE(2)$-convolutions and the Fourier transform on $SE(2)$, which makes apparent that our steerable $SE(2)$-convolution is faster since a number of unnecessary calculations are omitted. Finally, it is demonstrated how $SE(2)$-convolutions can be used to calculate stochastic completion fields for the purpose of line completion.

In Chapter 4, we explain that $SE(2)$-convolutions can be applied to tensor image processing methods as well, which is derived for *tensor voting* and *structure tensor* methods. We propose to calculate the tensor voting operation in a steer-

able way, which is more efficient if tensor voting is applied on dense input data. Furthermore, we investigate the application of steerable tensor voting on electrophysiology catheter extraction, resulting in an increased extraction performance.

To compute well-posed left-invariant derivatives in orientation scores, we discuss the need for regularized left-invariant derivatives in Chapter 5, where left-invariant diffusion is the logical choice to use for the regularization. For the special case of spatially isotropic diffusion on $SE(2)$, we can use the implementations of ordinary Gaussian derivatives. In the other cases, we propose to calculate the regularization and the derivatives numerically. We use the regularized derivatives to estimate descriptive local features in orientation scores: curvature, deviation from horizontality, and orientation confidence. These features are estimated by finding the exponential curve that optimally fits to the local orientation score data. This optimal fit is found by using the left-invariant Hessian on the orientation score. The results show that the feature estimation works quite well and improves other approaches.

In Chapter 6 we use the results of the previous chapters to develop a nonlinear *coherence-enhancing diffusion method* for curve enhancement in images via orientation scores. The diffusion process is controlled by the estimated local features. We propose two explicit numerical schemes to operationalize this nonlinear diffusion process. The first scheme uses finite differences on the sampling grid, which is best concerning speed, and the second one uses interpolated left-invariant finite differences, which is best concerning rotation invariance. The coherence-enhancing diffusion method clearly enhances elongated structures while preserving crossing structures. This is a beneficial property for many biomedical applications, as is demonstrated on microscopy images of collagen fibres, muscle cells, and bone tissue.

Besides orientation scores of 2D images, we also investigate orientation scores of 3D images. In Chapter 7, all basic theory is mapped to the 3D case. However, one of the complicating factors is that most 3D orientation scores are not functions on the entire group $SE(3)$, but rather on a coset space of the group. We discuss basic implementations of 3D orientation score algorithms and we show that these algorithms can be applied to diffusion tensor imaging (DTI) data and especially high angular resolution diffusion imaging (HARDI) data.

Finally, the development of a mathematically-oriented image processing library in C++ is described in Chapter 8. This library is intended to be applicable to many image processing research projects and applications. By extensive use of *C++ template* constructions, the library is designed to be as generic and flexible as possible while at the same time computationally efficient. Furthermore, to combine the advantages of both C++ and Mathematica, the C++ library can be used conveniently in conjunction with Mathematica in an efficient way using a Mathlink caching scheme. The algorithms described in this thesis are incorporated into this library.

9.2 Future Research

In this final section we discuss potentially interesting topics for future research related to the work described in this thesis.

This thesis focussed on enhancement of elongated structures. Next to enhancement, orientation scores also provide a useful framework for curve *detection*, which is often demanded, for instance for vessel tree segmentation. It is an interesting branch of further research to develop detection methods using orientation scores, which take advantage of the enhancement capabilities of the algorithms described in this thesis, in order to improve the performance compared to traditional detection approaches.

Furthermore, it would be of interest to make *scale* an explicit dimension as well, to improve enhancement of images with both thin and thick line structures, for instance retina images [126]. A 2D image will render a 4D scale-orientation score, which is a function on the *similitude group*.

Due to the noncommutative structure of the Lie algebra of $SE(2)$, applying diffusion in the orientation dimension also leads to some diffusion in the direction orthogonal to elongated structures. This can leads to undesired blurring of elongated structures. The introduction of a "thinning" mechanism to counteract for blurring orthogonal to line structures is a promising approach to obtain sharper results. Such thinning mechanisms can be accomplished by adaptive morphological operations on orientation scores, which amount to adding adaptive convection terms to the partial differential equation being solved. A challenging topic in this case is the development of numerical schemes for the nonlinear convection part.

Finally, many (bio)medical applications demand orientation scores of 3D images. Therefore, it is definitely worthwhile continuing research on 3D orientation scores, for which Chapter 7 of this thesis is a starting point. Practically oriented research might include the application of the methods on real HARDI data and the development of efficient implementations of $SE(3)$-convolutions and $SE(3)$-diffusions, either on CPU, GPU, or FPGA. More theoretical research topics include the derivation of analytic formulas of Green's functions for diffusion (and convection) equations on $SE(3)$.

Bibliography

[1] David Abrahams and Aleksey Gurtovoy. *C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond*. C++ In-depth. Addison-Wesley Professional, 2004.

[2] M. A. van Almsick. *Context Models of Lines and Contours*. PhD thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, 2007.

[3] M. A. van Almsick, R. Duits, E.M. Franken, and B.M. ter Haar Romeny. From stochastic completion fields to tensor voting. In O. Fogh Olsen, L.M.J. Florack, and A. Kuijper, editors, *Deep Structure, Singularities and Computer Vision*, volume 3753 of *Lecture Notes in Computer Science*, pages 124–134. Springer-Verlag, 2005.

[4] J.-P. Antoine and R. Murenzi. Two-dimensional directional wavelets and the scale-angle representation. *Signal Processing*, 52(3):241–272, 1996.

[5] J.-P. Antoine, R. Murenzi, and P. Vandergheynst. Directional Wavelets Revisited: Cauchy Wavelets and Symmetry Detection in Patterns. *Applied and Computational Harmonic Analysis*, 6(3):314–345, 1999.

[6] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

[7] Jonas August. *The Curve Indicator Random Field*. PhD thesis, Yale University, December 2001.

[8] H. Bårman, G. H. Granlund, and H. Knutsson. A new approach to curvature estimation and description. In *3rd International Conference on Image Processing and its Applications*, pages 54–58, Warwick, Great Britain, July 1989. IEE.

[9] P. J. Basser, J. Mattiello, and D. Le Bihan. Estimation of the effective self-diffusion tensor from the NMR spin echo. *Journal of Magnetic Resonance*, 103:247–254, 1994.

[10] P. J. Basser, J. Mattiello, and D. Le Bihan. MR diffusion tensor spectroscopy and imaging. *Biophysics Journal*, 66(1):259–267, 1994.

[11] Guus Berenschot. 3d visualization of diffusion tensor imaging. Master's thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, May 2003. http://bmia.bmt.tue.nl/education.

[12] J.A.R. Blais, D.A. Provins, and M.A. Soofi. Optimization of spherical harmonic transform computations. In *Computational Science - ICCS 2005*, volume 3513 of *Lecture Notes in Computer Science*, pages 74–81. Springer-Verlag, 2005.

[13] J. Blom. *Topological and Geometrical Aspects of Image Structure*. PhD thesis, University of Utrecht, Department of Medical and Physiological Physics, Utrecht, The Netherlands, 1992.

[14] W. H. Bosking, Y. Zhang, B. Schofield, and D. Fitzpatrick. Orientation selectivity and the arrangement of horizontal connections in tree shrew striate cortex. *Journal of Neuroscience*, 6(17):2112–2127, March 15 1997.

[15] T. Brox, J. Weickert, B. Burgeth, and P. Mrázek. Nonlinear structure tensors. *Image and Vision Computing*, 24(1):41–55, January 2006.

[16] Stijn de Buck, Frederik Maes, Joris Ector, Jan Bogaert, Steven Dymarkowski, Hein Heidbchel, and Paul Suetens. An augmented reality system for patient-specific guidance of cardiac catheter ablation procedures. *IEEE Transactions on Medical Imaging*, 24(11):1512–1524, November 2005.

[17] B. Burgeth, S. Didas, L. M. J. Florack, and J. Weickert. Nonlinear PDEs for matrix-fields. Presentation at the workshop on Variational and PDE Level Set Methods, Obergugl, Austria, September 1–3 2006.

[18] E.J. Candès and D.L. Donoho. Curvelets - a surprisingly effective nonadaptive representation for objects with edges. In A. Cohen, C. Rabut, and L.L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 1999, Nashville, TN*. Vanderbilt University Press, 1999.

[19] E.J. Candès and D.L. Donoho. Ridgelets: the key to high dimensional intermittency? *Philosophical Transactions of the Royal Society of London A*, 357:2495–2509, 1999.

[20] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of the International Conference on Image Processing 1994 (Austin, Switzerland, November 13–16, 1994)*, volume 2, pages 168–172. IEEE Computer Society Press, 1994.

[21] J. Chen, Y. Sato, and S. Tamura. Orientation space filtering for multiple orientation line segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(5):417–429, May 2000.

[22] Gregory S. Chirikjian and Alexander B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis*. CRC Press, 2001.

[23] G. Citti and A. Sarti. A cortical based model of perceptional completion in the roto-translation space. *Journal of Mathematical Imaging and Vision*, 24(3):307–326, 2006.

[24] Doxygen Source code documentation generator tool. http://www.doxygen.org/.

[25] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[26] G.-H. Cottet and L. Germain. Image processing through reaction combined with nonlinear diffusion. *Mathematics of Computation*, 61:659–667, 1993.

[27] Renske de Boer. Perceptual grouping: The closure of gaps within elongated structures in medical images. Master's thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, March 2006. Internal Report number BMIA 06/02, http://bmia.bmt.tue.nl/education".

[28] M. Descoteaux, E. Angelino, S. Fitzgibbons, and R. Deriche. Regularized, fast, and robust analytical Q-ball imaging. *Magnetic Resonance in Medicine*, 58(3):497–510, 2007.

[29] DIPimage and DIPlib. http://www.diplib.org/.

[30] P.A.G. van Dorst. Motion extraction based on multiscale anchor point movement and a soft constraint for greyscale conservation. Master's thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, The Netherlands, 2007. http://bmia.bmt.tue.nl/education.

[31] J.R. Driscoll and D.M. Healy Jr. Computing Fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15(2):202–250, 1994.

[32] DTI Tool. http://www.bmia.bmt.tue.nl/software/dtitool/.

[33] R. Duits. *Perceptual Organization in Image Analysis*. PhD thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, 2005. http://www.bmi2.bmt.tue.nl/Image-Analysis/People/RDuits/THESISRDUITS.pdf.

[34] R. Duits, M. A. van Almsick, M. Duits, E.M. Franken, and L.M.J. Florack. Image processing via shift-twist invariant operations on orientation bundle functions. In Niemann Zhuralev et al., editors, *7th International Conference on Pattern Recognition and Image Analysis (PRIA-7-2004), St. Petersburg*, pages 193–196, October 2004.

[35] R. Duits and M.A. van Almsick. The explicit solutions of linear left-invariant second order stochastic evolution equations on the 2D-Euclidean motion group. *AMS Quarterly of Applied Mathematics*, 66:27, 2008.

[36] R. Duits, M. Duits, M.A. van Almsick, and B.M. ter Haar Romeny. Invertible orientation scores as an application of generalized wavelet theory. *Pattern Recognition and Image Analysis (PRIA)*, 17(1):42–75, 2007.

[37] R. Duits, M. Felsberg, G. Granlund, and B. M. ter Haar Romeny. Image analysis and reconstruction using a wavelet transform constructed from a reducible representation of the Euclidean motion group. *International Journal of Computer Vision*, 72(1):79–102, April 2007.

[38] R. Duits, L. Florack, J. de Graaf, and B. M. ter Haar Romeny. On the axioms of scale space theory. *Journal of Mathematical Imaging and Vision*, 20(3):267–298, 2004.

[39] R. Duits and E.M. Franken. Left-invariant parabolic evolutions on $SE(2)$ and contour enhancement via invertible orientation scores - part I: Linear left-invariant diffusion equations on $SE(2)$. Submitted to the Quarterly on Applied mathematics, AMS, 2008.

[40] R. Duits and E.M. Franken. Left-invariant parabolic evolutions on $SE(2)$ and contour enhancement via invertible orientation scores - part II: Nonlinear left-invariant diffusions on invertible orientation scores. Submitted to the Quarterly on Applied mathematics, AMS, 2008.

[41] R. Duits, H Führ, and B.J. Janssen. Left-invariant evolution equation on Gabor transforms, 2008. In preparation for submission to AMS.

[42] F.G.A. Faas and L.J. van Vliet. 3D-orientation space; filters and sampling. In J. Bigun and T. Gustavsson, editors, *Proc. 13th Scandinavian Conference on Image Analysis (SCIA)*, volume 2749 of *Lecture Notes in Computer Science*, page 36, 2003.

[43] Daniel Fagerström. Spatio-temporal scale-spaces. In Sgallari et al. [123], pages 326–337.

[44] P. Fallavollita, P. Savard, and G Sierra. Fluoroscopic navigation to guide RF catheter ablation of cardiac arrhythmias. In *26th Annual International Conference of the Engineering in Medicine and Biology Society*, 2004.

[45] M. Felsberg, P.-E. Forssén, and H. Scharr. Channel smoothing: Efficient robust smoothing of low-level signal features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):209–222, 2006.

[46] L. M. J. Florack. *Image Structure*, volume 10 of *Computational Imaging and Vision Series*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

[47] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Cartesian differential invariants in scale-space. *Journal of Mathematical Imaging and Vision*, 3(4):327–348, November 1993.

[48] L.M.J. Florack. Codomain scale space and regularization for high angular resolution diffusion imaging. *Computer Vision and Pattern Recognition Workshops, 2008. CVPR Workshops 2008. IEEE Computer Society Conference on*, pages 1–6, June 2008.

[49] J. Foolen, C. van Donkelaar, N. Nowlan, P. Murphy, R. Huiskes, and K. Ito. Collagen orientation in periosteum and perichondrium is aligned with preferential directions of tissue growth. *Journal of orthopaedic research: official publication of the Orthopaedic Research Society*, 2008.

[50] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS Intercommission Conf. on Fast Processing of Photogrammetric Data (Interlaken)*, pages 281–305, 1987.

[51] E. M. Franken, M.A. van Almsick, P. Rongen, L. M. J. Florack, and B. M. ter Haar Romeny. An efficient method for tensor voting using steerable filters. In Leonardis et al. [90], pages 228–240.

[52] E.M. Franken. Context-enhanced detection of electrophysiology catheters in noisy fluoroscopy images. Master's thesis, Technische Universiteit Eindhoven, Department of Electrical Engineering and Department of Biomedical Engineering, The Netherlands, 2004. http://bmia.bmt.tue.nl/education.

[53] E.M. Franken and R. Duits. Crossing-preserving coherence-enhancing diffusion on invertible orientation scores. Accepted for publication in the International Journal of Computer Vision (IJCV).

[54] E.M. Franken, R. Duits, and B. M. ter Haar Romeny. Nonlinear diffusion on the 2D Euclidean motion group. In Sgallari et al. [123], pages 461–472.

[55] W. Freeden and M. Schreiner. Non-orthogonal Expansions on the Sphere. *Mathematical Methods in the Applied Sciences*, 18:83–120, 1995.

[56] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

[57] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

[58] S. Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Mat. Nauk 7*, pages 749–754, 1931.

[59] M. van Ginkel, J. van de Weijer, L.J. van Vliet, and P.W. Verbeek. Curvature estimation from orientation fields. *Proceedings of the 11th Scandinavian Conference on Image Analysis (Kangerlussuaq, Greenland, June 7–11 1999)*, 2:545–552, 1999.

[60] G. H. Granlund. An associative perception-action structure using a localized space variant information representation. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

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

[62] K. Gröchenig. *Foundations of Time-Frequency Analysis*. Birkhauser, 2001.

[63] Y. Gur and N. Sochen. Fast invariant riemannian DT-MRI regularization. *Proceedings of the 11th International Conference on Computer Vision (Rio de Janeiro, Brazil, October 14–20, 2007)*, pages 1–7, Oct. 2007.

[64] Gideon Guy and Gérard Medioni. Inferring global perceptual contours from local features. *International Journal of Computer Vision*, 20(1–2):113–33, October 1996.

[65] A. Haar. Der Massbegriff in der Theorie der kontinuierlichen Gruppen. *Ann. of Math*, 34:147–169, 1933.

[66] B. M. ter Haar Romeny. *Front-End Vision and Multi-Scale Image Analysis: Multi-Scale Computer Vision Theory and Applications, written in Mathematica*, volume 27 of *Computational Imaging and Vision Series*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003.

[67] B. M. ter Haar Romeny, B. Titulaer, S.N. Kalitzin, G. Scheffer, F. Broekmans, J.J. Staal, and E. te Velde. Computer assisted human follicle analysis for fertility prospects with 3D ultrasound. In A. Kuba, M. Sámal, and A. Todd-Pokropek, editors, *Information Processing in Medical Imaging*, volume 1613 of *Lecture Notes in Computer Science*, page 5669, Heidelberg, 1999. Springer-Verlag.

[68] Morton Hamermesh. *Group theory and its application to physical problems*. Addison-Wesley, 1962.

[69] F. Heitger and R. von der Heydt. A computational model of neural contour processing. In *Proceedings of the 4th International Conference on Computer Vision (Berlin, Germany, June 20–23, 1993)*, pages 32–40. Washington D.C.: IEEE Computer Society Press, 1993.

[70] Y. Hel-Or and C. Teo, P. A common framework for steerability, motion estimation, and invariant feature detection. *Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on*, 5:337–340.

[71] M. Hlawitschka, J. Ebling, and G. Scheuermann. Convolution and Fourier transform of second order tensor fields. In J.J. Villanieva, editor, *Visualization, Imaging, and Image Processing (VIIP)*, 2004.

[72] D. H. Hubel. *Eye, Brain and Vision*, volume 22 of *Scientific American Library*. Scientific American Press, New York, 1988.

[73] Gijs Huisman. Perceptual grouping: Curvature enhanced closure of elongated structures. Master's thesis, Technische Universiteit Eindhoven, Department of Biomedical Engineering, The Netherlands, July 2006. Report number BMIA 07/07, http://bmia.bmt.tue.nl/education".

[74] T. Iijima. Basic theory of pattern observation. Papers of Technical Group on Automata and Automatic Control, 1959. (in Japanese).

[75] Mariusz Jankowski. Digital Image Processing package. http://www.wolfram.com/products/applications/digitalimage/.

[76] B. Janssen, R. Duits, and B. M. ter Haar Romeny. Linear image reconstruction by Sobolev norms on the bounded domain. In Sgallari et al. [123], pages 55–67.

[77] B. J. Janssen, L. M. J. Florack, R. Duits, and B. M. ter Haar Romeny. Optic flow from multi-scale dynamic anchor point attributes. In A. Campilho and M. Kamel, editors, *ICIAR 2006: Proceedings of the Third International Conference on Image Analysis and Recognition, Póvoa de Varzim, Portugal*, volume 4141 of *Lecture Notes in Computer Science*, pages 767–779, Berlin, September 2006. Springer-Verlag.

[78] B. J. Janssen, F. M. W. Kanters, R. Duits, L. M. J. Florack, and B. M. ter Haar Romeny. A linear image reconstruction framework based on Sobolev type inner products. *International Journal of Computer Vision*, 70(3):231–240, December 2006.

[79] B. Jian, B. C. Vemuri, E. Özarslan, P. R. Carney, and T. H. Mareci. A novel tensor distribution model for the diffusion-weighted MR signal. *NeuroImage*, 37:164–176, 2007.

[80] S. N. Kalitzin, B. M. ter Haar Romeny, and M. A. Viergever. Invertible apertured orientation filters in image analysis. *International Journal of Computer Vision*, 31(2/3):145–158, April 1999.

[81] S.N. Kalitzin, B. M. ter Haar Romeny, and M.A. Viergever. Invertible orientation bundles on 2D scalar images. In B. M. ter Haar Romeny, L.M.J. Florack, J. Koenderink, and M. Viergever, editors, *Scale Space Theory in Computer Vision*, pages 77–88, 1997.

[82] K. Kanatani. *Group-Theoretical Methods in Image Understanding*, volume 20 of *Series in Information Sciences*. Springer-Verlag, 1990.

[83] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, fourth edition, 2000.

[84] Morton J. Kern. *The Cardiac Catheterization Handbook*. Mosby, fourth edition, 2003.

[85] G. Kin and M. Sato. Scale space filtering on spherical pattern. *Proc. 11th international conference on Pattern Recognition, vol. C*, pages 638–641, 1992.

[86] U. Köthe. Edge and junction detection with an improved structure tensor. In B. Michaelis and G. Krell, editors, *Pattern Recognition, Proc. of 25th DAGM Symposium, Magdeburg 2003*, volume 2781 of *Lecture Notes in Computer Science*, pages 25–32. Springer-Verlag, 2003.

[87] S. Kunis and D. Potts. Fast spherical Fourier algorithms. *Journal of Computational and Applied Mathematics*, 161(1):75–98, 2003.

[88] A.B. Kyatkin and G.S. Chirikjian. Algorithms for fast convolutions on motion groups. *Applied and Computational Harmonic Analysis*, 9(2):220–241, 2000.

[89] David B. Kynor, Anthony J. Dietz, Eric M. Friets, Jon N. Peterson, Ursula C. Bergstrom, John K. Triedman, and Peter E. Hammer. Visualization of cardiac

wavefronts using data fusion. In Milan Sonka, editor, *Medical imaging 2002: Image Processing. Proceedings of the SPIE*, volume 4684, pages 1186–1194. SPIE-Int. Soc. Opt. Eng, 2002.

[90] A. Leonardis, H. Bischof, and A. Prinz, editors. *Proceedings of the Ninth European Conference on Computer Vision (Graz, Austria, May 2006)*, volume 3951–3954 of *Lecture Notes in Computer Science*, Berlin, Heidelberg, 2006. Springer-Verlag.

[91] Boost C++ Libraries. http://www.boost.org/.

[92] CMake Cross-Platform Make. http://www.cmake.org/.

[93] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1999.

[94] R. Manniesing and W.J. Niessen. Multiscale vessel enhancing diffusion in ct angiography noise filtering. In *Information Processing in Medical Imaging*, volume 3565 of *Lecture Notes in Computer Science*, pages 138–149, Berlin, 2005. Springer-Verlag.

[95] R. Manniesing, M. A. Viergever, and W.J. Niessen. Vessel enhancing diffusion: A scale space representation of vessel structures. *Medical Image Analysis*, 10(6):815–825, 2006.

[96] MathVisionTools. http://www.mathvisiontools.net/.

[97] MathWorks MatLab Image Processing Toolbox. http://www.mathworks.com/products/image/.

[98] Gérard Medioni, Mi-Suen Lee, and Chi-Keung Tang. *A Computational Framework for Segmentation and Grouping*. Elsevier Science, 2000.

[99] Gérard Medioni and Philippos Mordohai. *Emerging Topics in Computer Vision*, chapter 5: "The Tensor Voting Framework", pages 191–252. IMSC Press Multimedia Series. Prentice-Hall, 2004.

[100] M. Michaelis and G. Sommer. A Lie group approach to steerable filters. *Pattern Recognition Letters*, 16:1165–1174, 1995.

[101] Chikara Miyaji and Paul Abbott. *MathLink: Network Programming with Mathematica*. Cambridge University Press, May 2001.

[102] S. Mori. *Introduction to Diffusion Tensor Imaging*. Elsevier Science, 2007.

[103] D. Mumford. Elastica and computer vision. In C. L. Bajaj, editor, *Algebraic Geometry and its Applications*, pages 491–506. Springer-Verlag, New York, 1994.

[104] M. Nitzberg and T. Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:826–833, 1992.

[105] K. Nordberg, G. Granlund, and H. Knutsson. Representation and Learning of Invariance. In *Proceedings of IEEE International Conference on Image Processing*, Austin, Texas, November 1994. IEEE.

[106] K. Nordberg, H. Knutsson, and G. Granlund. Local curvature from gradients of the orientation tensor field. Report LiTH-ISY-R-1783, Computer Vision Laboratory, SE-581 83 Linköping, Sweden, August 1995.

[107] Open CV library. http://www.intel.com/technology/computing/opencv/.

[108] E. Özarslan and T. H. Mareci. Generalized diffusion tensor imaging and analytical relationships between diffusion tensor imaging and high angular resolution imaging. *Magnetic Resonance in Medicine*, 50:955–965, 2003.

[109] E. Özarslan, T. M. Shepherd, B. C. Vemuri, S. J. Blackband, and T. H. Mareci. Resolution of complex tissue microarchitecture using the diffusion orientation transform (DOT). *NeuroImage*, 31:1086–1103, 2006.

[110] T.H.J.M. Peeters, A. Vilanova, G.J. Strijkers, and B.M. ter Haar Romeny. Visualization of the Fibrous Structure of the Heart. *Vision, Modeling, and Visualization 2006: Proceedings, November 22-24, 2006, Aachen, Germany*, 2006.

[111] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[112] B. Platel, E. G. Balmachnova, L. M. J. Florack, and B. M. ter Haar Romeny. Top-points as interest points for image matching. In Leonardis et al. [90], pages 418–429.

[113] V. Prckovska, A.F. Roebroeck, P. Pullen, A. Vilanova, and B.M. ter Haar Romeny. Optimal acquisition schemes in high angular diffusion imaging. In *Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention—MICCAI 2006 (New York, USA, September 6-10, 2008)*, 2008.

[114] Intel Integrated Performance Primitives. http://www.intel.com/cd/software/products/asmo-na/eng/perflib/ipp/302910.htm.

[115] ImageJ Image Processing and Analysis in Java. http://rsbweb.nih.gov/ij/.

[116] Richard van der Put. Methods for 3D orientation analysis and their application to the study of arterial remodeling. Master's thesis, Technische Universiteit Eindhoven, March 2006. Interal Report number BMIA 05/02.

[117] X. Qian, M.P. Brennan, D.P. Dione, W.L. Dobrucki, M.P. Jackowski, C.K. Breuer, A.J. Sinusas, and X. Papademetris. Detection of Complex Vascular Structures using Polar Neighborhood Intensity Profile. *Proceedings of the 11th International Conference on Computer Vision (Rio de Janeiro, Brazil, October 14–20, 2007)*, pages 1–8, 2007.

[118] M. Reisert and H. Burkhardt. Efficient Tensor Voting with 3D Tensorial Harmonics. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2008, Workshop on Tensors.*, pages 1–7, 2008.

[119] M.P. Rubbens, A. Mol, R.A. Boerboom, R.A. Bank, F.P.T. Baaijens, and C.V.C. Bouten. Intermittent Straining Accelerates the Development of Tissue Properties in Engineered Heart Valve Tissue. *Tissue Engineering Part A*. ahead of print. doi:10.1089/ten.tea.2007.0396.

[120] B. Saff and A.B.J. Kuijlaars. Distributing many points on a sphere. *the Mathematical Intelligencer*, 19(1):5, 1997.

[121] H. Scharr. Diffusion-like reconstruction schemes from linear data models. In *Pattern Recognition : 28th DAGM Symposium, Berlin, Germany, September 12-14*, volume 4174 of *Lecture Notes in Computer Science*, pages 51–60. Springer-Verlag, 2006.

[122] ITK segmentation and registration toolkit. http://www.itk.org/.

[123] F. Sgallari, A. Murli, and N. Paragios, editors. *Scale Space and Variational Methods in Computer Vision: Proceedings of the First International Conference, SSVM 2007, Ischia, Italy*, volume 4485 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, May–June 2007.

[124] C.S. Shaw, D.A. Jones, and A.J.M. Wagenmakers. Network distribution of mitochondria and lipid droplets in human muscle fibres. *Histochemistry and Cell Biology*, 129(1):65–72, 2008.

[125] J. Sporring, M. Nielsen, L. Florack, and P. Johansen. *Gaussian scale-space theory*. Kluwer Academic Publishers, 1997.

[126] J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004.

[127] Jean-Luc Starck, Emmanuel J. Candès, and David L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):670–684, June 2002.

[128] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.

[129] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, 2000.

[130] I. Stuke, T. Aach, E. Barth, and C. Mota. Analysing superimposed oriented patterns. In *6th IEEE SSIAI*, pages 133–137, 2004.

[131] C++ template image processing toolkit. http://cimg.sourceforge.net/.

[132] P. Teo and Y. Hel-Or. A computational group-theoretic approach to steerable functions. *Pattern Recognition Letters*, 19(1):7–17, 1998.

[133] Wai-Shun Tong, Chi-Keung Tang, Philippos Mordohai, and Gérard Medioni. First order augmentation to tensor voting for boundary inference and multiscale analysis in 3D. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):594–611, 2004.

[134] D. Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *International Journal of Computer Vision*, 68(1):65, 2006.

[135] D. Tschumperlé and R. Deriche. Diffusion tensor regularization with constraints preservation. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, 1:948–953, 2001.

[136] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999.

[137] Ginkel. M. van. *Image Analysis using Orientation Space based on Steerable Filters*. PhD thesis, Technische Universiteit Delft, The Netherlands, 2002.

[138] M. van Ginkel, PW Verbeek, and LJ van Vliet. Curvature Estimation for Overlapping Curved Patterns using Orientation Space. *Proceedings of the third annual conference of the Advanced School for Computing and Imaging, Lommel, Belgium*, pages 173–178, 1998.

[139] V.S. Varadarajan. *Lie groups, Lie algebras, and their representations*. Prentice-Hall, 1974.

[140] Todd L. Veldhuizen. C++ templates are Turing complete. http://ubiety.uwaterloo.ca/~tveldhui/papers/2003/turing.pdf, 2003.

[141] P.W. Verbeek, L.J. van Vliet, and J. van de Weijer. Improved curvature and anisotropy estimation for curved line bundles. *Proceedings of the 14th International Conference on Pattern Recognition (Brisbane, Australia, August 16–20, 1998)*, 1:528–533, 1998.

[142] A. Vilanova, S. Zhang, G. Kindlmann, and D. Laidlaw. *Visualization And Processing of Tensor Fields*, chapter An introduction to visualization of diffusion tensor imaging and its applications, pages 121–153. Springer, 2006.

[143] Deborah Walters. Selection of image primitives for general-purpose visual processing. *Computer Vision, Graphics, and Image Processing*, 37(2):261–298, 1987.

[144] J. Weickert and H. Scharr. A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance. *Journal of Visual Communication and Image Representation*, pages 103–118, 2002.

[145] J. A. Weickert. *Anisotropic Diffusion in Image Processing*. PhD thesis, University of Kaiserslautern, Department of Mathematics, Kaiserslautern, Germany, January 29 1996.

[146] J. A. Weickert. *Anisotropic Diffusion in Image Processing*. ECMI Series. Teubner, Stuttgart, January 1998.

[147] J. A. Weickert. Coherence-enhancing diffusion filtering. *International Journal of Computer Vision*, 31(2/3):111–127, April 1999.

[148] J. van de Weijer, L.J. van Vliet, P.W. Verbeek, and M. van Ginkel. Curvature estimation in oriented patterns using curvilinear models applied to gradient vector fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1035–1042, 2001.

[149] Martin Welk, Gabriele Steidl, and Joachim Weickert. Locally analytic schemes: A link between diffusion filtering and wavelet shrinkage. *Applied and Computational Harmonic Analysis*, 24(2):195–224, March 2008. Special Issue on Mathematical Imaging.

[150] Martin Welk, Joachim Weickert, and Gabriele Steidl. From tensor-driven diffusion to anisotropic wavelet shrinkage. In Leonardis et al. [90], pages 391–403.

[151] Lance R. Williams and David W. Jacobs. Stochastic completion fields: a neural model of illusory contour shape and salience. *Neural Comput.*, 9(4):837–858, 1997.

[152] A. P. Witkin. Scale-space filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, Germany, 1983.

[153] Wolfram Mathematica. http://www.wolfram.com/products/mathematica/.

[154] Anna Yershova and Steven M. LaValle. Deterministic sampling methods for spheres and SO(3). In *IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[155] John Zweck and Lance R. Williams. Euclidean group invariant computation of stochastic completion fields using shiftable-twistable functions. *Journal of Mathematical Imaging and Vision*, 21(2):135–154, September 2004.

Samenvatting

Voor veel medische en biomedische applicaties is het verbeteren van de zichtbaarheid van langgerekte structuren in ruizige beelden van groot belang. De huidige algemeen toepasbare methoden in de beeldbewerkingsliteratuur hebben echter problemen met het verwerken van *kruisende* langgerekte structuren. In dit proefschrift stellen we voor om dit probleem op te lossen door gebruik te maken van zogenaamde *oriëntatie partituren*. Het idee hiervan is dat de verschillende oriëntaties in het beeld worden gescheiden zodat ze onafhankelijk van elkaar behandelt kunnen worden. Een oriëntatie partituur van een twee-dimensionaal beeld is dus een drie-dimensionale ruimte, waarin de oriëntatie de derde dimensie vormt. Door ervoor te zorgen dat uit de oriëntatie partituur het originele beeld kan worden gereconstrueerd, kunnen we een beeld verbeteren of analyseren *via* de oriëntatie partituur van het beeld.

In Hoofdstuk 2 wordt de noodzakelijke wiskunde met betrekking tot oriëntatie partituren geïntroduceerd. We beschrijven hoe een oriëntatie partituur wordt geconstrueerd door een twee-dimensionaal beeld te convolueren met een roterende anisotrope filter-kernel. De transformatie is inverteerbaar indien alle geroteerde filter-kernels samen alle relevante frequentie componenten bevatten. Oriëntatie partituren kunnen worden beschouwd als functies op de Euclidische bewegingsgroep, genoteerd als $SE(2)$. Deze observatie maakt het mogelijk om veel bekende methoden in de beeldbewerking te "vertalen" naar het domein $SE(2)$ van een oriëntatie partituur. Om correcte operaties te definiëren op oriëntatie partituren dient de $SE(2)$-groepsstructuur in acht genomen te worden. We geven daarom een korte introductie in de Lie groepen theorie en tonen vervolgens aan dat als de effectieve operaties op een beeld *translatie- en rotatie-invariant* moeten zijn, hetgeen meestal wenselijk is, de corresponderende operatie op de oriëntatie partituur *links-invariant* dient te zijn. Derhalve zorgen we ervoor dat alle operaties links-invariant zijn en beginnen we met de afleiding van links-invariante partiële afgeleiden. Daarna introduceren we kort de differentiaalgeometrie op $SE(2)$, die belangrijk is om lokale structuren in oriëntatie partituren te beschrijven. We introduceren exponentiële curven in $SE(2)$, en laten zien dat dit spiralen zijn die equivalent zijn aan rechte lijnen in een Euclidische ruimte $\mathbb{R}^n$. Tenslotte introduceren we links-invariant diffusievergelijkingen en convectie-diffusievergelijkingen voor functies op $SE(2)$, waarop de meeste bewerkingen op oriëntatie partituren in dit proefschrift zijn gebaseerd.

In Hoofdstuk 3 beschouwen we vervolgens lineaire links-invariante operaties op oriëntatie partituren. Deze subklasse van operaties kan door middel van de $SE(2)$-*convolutie* worden geoperationaliseerd. We laten zien dat een dergelijke convolutie kan worden uitgevoerd door middel van *stuurbare filters*. Dit leidt tot efficiëntere implementaties indien de $SE(2)$-filter-kernel voldoende bandbegrensd is met betrekking tot de hoek. We leiden ook het verband af tussen stuurbare $SE(2)$-

convoluties en de Fourier transformatie op $SE(2)$. Hiermee laten we zien dat de voorgestelde implementatie voor stuurbare $SE(2)$-convoluties sneller is omdat een aantal berekeningen niet hoeft te worden uitgevoerd. Tenslotte demonstreren we hoe $SE(2)$-convoluties gebruikt kunnen worden om *stochastische completeringsvelden* te berekenen ten behoeve van het completeren van onderbroken lijnen in beelden.

In Hoofdstuk 4 leggen we uit dat $SE(2)$-convoluties ook kunnen worden toegepast bij de bewerking van tensor-waardige beelden. De verbanden met de *tensor voting* methode en de *structuur tensor* worden expliciet vastgesteld. Verder stellen we een methode voor om tensor voting op een stuurbare manier uit te voeren, hetgeen efficiënter is als tensor voting op dense data wordt uitgevoerd. Tenslotte bekijken we de toepassing van stuurbare tensor voting op de detectie van electrofysiologie catheters, hetgeen resulteert in in verbeterde resultaten.

Om goed-gestelde links-invariante afgeleiden in oriëntatie partituren de berekenen, stellen we in Hoofdstuk 5 voor om dergelijke afgeleiden te construeren door links-invariante diffusie voor de regularisatie te gebruiken. In het speciale geval van spatiëel isotrope diffusie kunnen we de implementatie van gewone Gaussische afgeleiden gebruiken. In andere gevallen is de meest praktische benadering om de benodigde $SE(2)$-convolutie-kernel numeriek te berekenen. We gebruiken de geregulariseerde afgeleiden om beschrijvende lokale eigenschappen in oriëntatie partituren te meten: kromming, afwijking van horizontaliteit, en geörienteerdheid. Deze eigenschappen worden geschat door de exponentiële curve te vinden die het best past bij de lokale structuur in de oriëntatie partituur. Deze optimale pasvorm wordt gevonden middels de links-invariante Hessiaan op de oriëntatie partituur. De resultaten tonen aan dat de meting van de lokale eigenschappen goed werkt en andere methoden verbetert.

In Hoofdstuk 6 gebruiken we de resultaten van de voorgaande hoofdstukken om een *niet-lineaire coherentie-verbeterende diffusie* methode te ontwikkelen voor het verbeteren van de zichtbaarheid van langgerekte structuren in beelden via oriëntatie partituren. Het diffusie proces wordt aangestuurd door geschatte lokale eigenschappen in de oriëntatie partituur. We beschrijven twee expliciete numerieke schema's om het niet-lineaire diffusie process te operationaliseren. Het eerste schema gebruikt eindige differenties direct op het bemonsteringsrooster, hetgeen leidt tot een snel algoritme. Het tweede schema, gebruikt geïnterpoleerde links-invariante eindige differenties, hetgeen beter is met betrekking tot rotatie-invariantie. Onze methode maakt langgerekte structure beter zichtbaar terwijl kruisingen behouden blijven. Dit is een voordeel voor veel biomedische applicaties, zoals wordt gedemonstreerd op microscopie beelden van collageen vezels, spiercellen en botweefsel.

Naast oriëntatie partituren van 2D beelden, onderzoeken we ook oriëntatie partituren van 3D beelden. In Hoofdstuk 7 wordt alle theorie "vertaald" naar het 3D geval. Echter, een van de complicerende factoren is dat de meeste 3D oriëntatie

partituren in de praktijk geen functies zijn op de gehele 3D Euclidische bewegingsgroep, maar enkel op een ruimte van nevenklassen van deze groep. Dit zorgt ervoor dat we extra zware eisen moeten opleggen aan de operaties die we mogen uitvoeren op de 3D oriëntatie partituur. Tenslotte beschrijven we eenvoudige implementaties voor 3D oriëntatie partituur algoritmes en tonen aan dat deze algoritmes kunnen worden toegepast high-angular resolution diffusion imaging (HARDI) beelden.

Tenslotte beschrijven we in Hoofdstuk 8 de ontwikkeling van een wiskundig georiënteerde beeldbewerkingsbibliotheek in C++. Deze bibliotheek is ontworpen om naast efficiënt, zo generiek en flexibel mogelijk te zijn. Aangezien in ons onderzoek ook veel gebruik is gemaakt van Mathematica, is het mogelijk gemaakt om de C++ bibliotheek op een eenvoudige en efficiënte manier samen met Mathematica te gebruiken. De algoritmes die in dit proefschrift zijn beschreven zijn in deze bibliotheek ondergebracht.

Dankwoord

Pffffoe. Nu ik dit schrijf is *het* AF! Vandaag gaat het dan eindelijk naar de drukker! En dit boekje is *het* dan geworden. Voor velen (maar hopelijk niet iedereen) oogt het waarschijnlijk slechts als een hoop pagina's vol met adacadabra en wat plaatjes... Maar het produceren van dit alles kostte toch 4 jaar aan "noeste arbeid": papers schrijven, papers lezen, wiskundige formules uitwerken, programmeren, experimenteren, conferenties bezoeken, enzovoorts. Al die zaken vormen ingrediënten voor dit proefschrift... Maar daarnaast was er nog veel meer nodig. Voor de overige "ingrediënten", die zeer essentieel waren om dit boekje tot stand te laten komen, heb ik deze 2 bladzijden ingeruimd.

Allereerst wil ik mijn copromotor Remco Duits bedanken. Remco, veel "fundamentalistische" theorie waarop dit proefschrift gebaseerd is komt van jouw hand. Ik als elektrotechnicus zou die theorie nooit volledig zelf hebben kunnen bedenken. Maar door jouw duidelijke uitleg kon ik er snel mee aan de gang om de theorie uit te breiden en vooral om het te "vertalen" naar bruikbare beeldbewerkingsmethoden. Jouw onbegrensde enthousiasme en ook vertrouwen in mijn werk waren een belangrijke bron van inspiratie om verder te gaan en steeds nieuwe zaken te onderzoeken. Jouw wiskundig niet accuraat te definiëren vorm van humor kan ik ook altijd erg waarderen. Kortom, bedankt! Mijn promotor Bart ter Haar Romeny wil ik ook van harte bedanken. Beste Bart, van de grote vrijheid die je me hebt gegeven heb ik veel geleerd, maar bovenal waardeer ik je grote enthousiasme en vertrouwen.

Daarnaast wil ik vele andere directe collega's bedanken. Markus van Almsick, bedankt voor de goede samenwerking, waaruit delen van Hoofdstuk 3 en 4 in dit proefschrift zijn voortgekomen. Luc Florack, bedankt voor het kritisch nakijken van veel van mijn manuscripten. Kamergenoot Bram, bedankt voor de autoventieldopjes en vele andere hulp. Voormalig kamergenoot Evgeniya, bedankt voor alle smalltalkjes en de schitterende foto's die je gemaakt hebt tijdens de diverse conferenties. En paranimf Bakel: Jooooooooow[1] Bakel, bedankt voor de prettige samenwerking met name bij lib. proggen, bedankt voor lenen van een doos Hertog ijs, en voor de biertjes in het walhalla, ... Verder bedank ik alle andere collega's en voormalig collega's bij BMIA die mijn promotietijd veraangenaamd hebben: Andrea, Alessandro, Anna, Ellen, Frans, Hans, Henri, Tim, Laura, Marieke, Neda, Paulo, Peter, Petr, Pieter, Ralph, Roy, en Vesna, de collega's bij modeling en alle collega's en ex-collega's die ik verder misschien vergeten ben. Ook de studenten die ik begeleid heb, Renske, Gijs, Justus, en Roel, ben ik dankbaar voor hun bijdragen. Bernard Burgeth, vielen dank for your hospitality during my stay in Saarbrücken. Een speciaal woord van dank ook aan voormalig-collega Arjen Ricksen, met wie ik na afloop van een conferentie een road trip in het midden van

[1] Spreek uit met zo laag mogelijke pitch.

de VS heb gemaakt. We hadden een ␣␣␣␣␣ tijd in de VS, en voor mij zijn zowel jouw aanwezigheid daar alsmede de megaranzige steaks zeker een katalysator geweest voor mijn “transformatie” ;-).

De lunch pauzes op de Odin kamer waren altijd erg gezellig, dus ook daar zijn er wat mensen die ik wil... (je raadt het nooit)... bedanken. Naast natuurlijk Bakel waren de vaste bezoekers aldaar: Iwan, Johan, en Thijs, ... Merci beaucoup voor het koffie zetten, discussies over van alles, en luisterende oren voor frustraties en stoom-afblazingen. Ook dank aan andere oud-studiegenoten die ik nog regelmatig zie, Marcel, Dirk, Jeroentje, Bakkus, Maes, Marjan, Sanne, Zef, Ronnie, en allen die ik vergeten ben. Oud huisgenoten Felan, Richard, Hans, Marco, Melchior, en Pim: was echt even wennen toen ik jullie na mijn verhuizing moest missen! Admar, aangezien ik jou al vanaf de kleuterschool ken en tot op heden altijd op dezelfde plek heb gezeten, vond ik het leuk om jou als paranimf te vragen. Bedankt voor alles al die jaren. Marc, bedankt dat je me er toe aangezet hebt om eens te gaan sporten, voor allerlei andere leukedingendoen, en voor de dikwijls verhelderende gesprekken over de meest uiteenlopende zaken die ons bezighouden. Ook bedankt aan Emmy, Heidi, Angelina, Patrick, Ronald, Tineke, Floortje, Killian en wie ik vergeten ben voor de leuke Ardennen weekendjes en diverse andere activiteiten.

Tenslotte, pa & ma, Carlijn & Batista, Laura & Marco bedankt voor al jullie gezelligheid en steun. Jeroen, idem en cool dat je helemaal vanuit Konstanz bent gekomen voor mijn promotie. Nichtjes Lisanne en Joline, een bezoek brengen aan jullie is altijd weer erg leutig. Ook bedankt aan alle andere familie... En Nicole, sinds ik jou ken is elke dag en vooral elk weekend een feest. Met jou was het afgelopen laatste-loodjes-jaar een stuk mooier. Bedankt++++!

Erik Franken
Eindhoven, 13 oktober 2008.

Curriculum Vitae

Erik Franken was born on the 16th of July 1979 in Roosendaal, The Netherlands. He graduated from the VWO (pre-university education) at the Norbertuscollege Roosendaal in 1998. In the same year he started studying Electrical Engineering at the Technische Universiteit Eindhoven, from which he graduated and received the M.Sc. degree cum laude in 2004. His graduation project involved the segmentation of Electrophysiology catheters in noisy fluoroscopy images, which was carried out in the Biomedical Image Analysis group at the Technische Universiteit Eindhoven and the X-ray predevelopment group at Philips Medical Systems Best. He continued his research within the Biomedical Image Analysis group as a Ph.D. student. The project focussed on the enhancement of crossing elongated structures in images.

Publications

- **Crossing-Preserving Coherence-Enhancing Diffusion on Invertible Orientation Scores.**
  E.M. Franken and R. Duits.
  Accepted for publication in International journal on computer vision (IJCV).

- **Left-Invariant Parabolic Evolutions on $SE(2)$ and Contour Enhancement via Invertible Orientation Scores - Part I: Linear Left-Invariant Diffusion Equations on $SE(2)$**
  R. Duits and E.M. Franken.
  Submitted to the Quarterly on Applied mathematics, AMS, 2008.

- **Left-Invariant Parabolic Evolutions on $SE(2)$ and Contour Enhancement via Invertible Orientation Scores - Part II: Nonlinear Left-Invariant Diffusions on Invertible Orientation Scores**
  R. Duits and E.M. Franken.
  Submitted to the Quarterly on Applied mathematics, AMS, 2008.

- **Curvature Estimation for Enhancement of Crossing Curves.**
  E. M. Franken, R. Duits, and B.M. ter Haar Romeny.
  In W. Niessen, C.-F. Westin, and M. Nielsen, editors, *Proceedings of the 8th IEEE Computer Society Workshop on Mathematical Methods in Biomedical Image Analysis (MMBIA), held in conjuction with the IEEE International Conference on Computer Vision (Rio de Janeiro, Brazil, October 14–20, 2007)*. Omnipress, 2007. Digital proceedings.
  Awarded the *MMBIA 2007 best paper award*.

- **Nonlinear Diffusion on the 2D Euclidean Motion Group.**
  E.M. Franken, R. Duits, and B.M. ter Haar Romeny.
  In F. Sgallari, A. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision: Proceedings of the First International Conference, SSVM 2007, Ischia, Italy*, volume 4485 of *Lecture Notes in Computer Science*, pages 461–472, Berlin, May–June 2007. Springer-Verlag.

- **Detection of Electrophysiology Catheters in Noisy Fluoroscopy Images.**
  E. M. Franken, P. Rongen, M.A. van Almsick, and B.M. ter Haar Romeny.
  In *Proceedings of the 9th International Conference on Medical Image Computing and Computer-Assisted Intervention—MICCAI 2006 (Copenhagen, Denmark, October 1-6, 2006)*, volume 4191 of *Lecture Notes in Computer Science*, pages 25–32, Berlin, 2006. Springer-Verlag.

- **An Efficient Method for Tensor Voting using Steerable Filters.**
  E. M. Franken, M.A. van Almsick, P. Rongen, L. M. J. Florack, and B.M. ter Haar Romeny.

In A. Leonardis, H. Bischof, and A. Prinz, editors, *Proceedings of the Ninth European Conference on Computer Vision (Graz, Austria, May 2006)*, volume 3951–3954 of *Lecture Notes in Computer Science*, pages 228–240, Berlin, Heidelberg, 2006. Springer-Verlag.

- **Visualization of Tooth Eruption in CT-Scans.**
  E.M. Franken, J. Sporring, and S. Kreiborg.
  In Tron Darvan, Nuno Hermann, Per Larsen, and Sven Kreiborg, editors, *Cranofacial Image Analysis for Biology, Clinical Genetics, Diagnostics and Treatment*, pages 33–39, October 2006. MICCAI 2006 Workshop Proceedings.

- **Steerable Tensor Voting.**
  E.M. Franken, M.A. van Almsick, P. Rongen, L.M.J. Florack, and B.M. ter Haar Romeny.
  In B. J. A. Kröse, H. J. Bos, E. A. Hendriks, and J. W. J. Heijnsdijk, editors, *Proceedings of the Eleventh Annual Conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands*, pages 65–72, June 8-10 2005.

- **From Stochastic Completion Fields to Tensor Voting.**
  M.A. van Almsick, R. Duits, E.M. Franken, and B.M. ter Haar Romeny.
  In O. Fogh Olsen, L.M.J. Florack, and A. Kuijper, editors, *Deep Structure, Singularities and Computer Vision*, volume 3753 of *Lecture Notes in Computer Science*, pages 124–134. Springer-Verlag, 2005.

- **Context-Enhanced Detection of Electrophysiology Catheters in X-Ray Fluoroscopy Images.**
  E.M. Franken, M.A. van Almsick, P.M.J. Rongen, B.M. ter Haar Romeny.
  European Congress on Radiology (ECR), Vienna, Austria, Conference Poster, 2005.

- **Context-Enhanced Detection of Electrophysiology Catheters in Noisy Fluoroscopy Images.**
  E.M. Franken.
  Master's thesis, Eindhoven University of Technology, Department of Electrical Engineering, The Netherlands, 2004.

- **Image Processing via Shift-Twist Invariant Operations on Orientation Bundle Functions.**
  R. Duits, M.A. van Almsick, M. Duits, E.M. Franken, and L.M.J. Florack.
  In Niemann Zhuralev et al., editors, *7th International Conference on Pattern Recognition and Image Analysis (PRIA-7-2004), St. Petersburg*, pages 193–196, October 2004.