An Efficient Method for Tensor Voting Using Steerable Filters

Erik Franken[1], Markus van Almsick[1], Peter Rongen[2], Luc Florack[1,⋆], and Bart ter Haar Romeny[1]

[1] Department of Biomedical Engineering, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{E.M.Franken, M.v.Almsick, L.M.J.Florack, B.M.terHaarRomeny}@tue.nl
[2] Philips Medical Systems, Best, The Netherlands
peter.rongen@philips.com

Abstract. In many image analysis applications there is a need to extract curves in noisy images. To achieve a more robust extraction, one can exploit correlations of oriented features over a spatial context in the image. Tensor voting is an existing technique to extract features in this way. In this paper, we present a new computational scheme for tensor voting on a dense field of rank-2 tensors. Using steerable filter theory, it is possible to rewrite the tensor voting operation as a linear combination of complex-valued convolutions. This approach has computational advantages since convolutions can be implemented efficiently. We provide speed measurements to indicate the gain in speed, and illustrate the use of steerable tensor voting on medical applications.

1 Introduction

Tensor voting (TV) was originally proposed by Guy and Medioni [1], and later presented in a book by Medioni et al. [2]. It is a technique for robust grouping and extraction of lines and curves from images. In noisy images, local feature measurements, i.e. measurements of local edges or ridges, are often unreliable, e.g. the curves are noisy and interrupted. TV aims at making these local feature measurements more robust by making them consistent with the measurements in the neighborhood. To achieve this, local image features strengthen each other if they are consistent according to a model for smooth curves.

TV is a an interesting and powerful method because of its simplicity and its wide applicability. However, the method exhibits some ad hoc concepts, namely the way the input data are encoded into a sparse tensor field representation, and the voting field model that is used. In this paper we focus on another problem: the current implementation is rather cumbersome in mathematical terms. A better mathematical formulation will help to better understand the method, and we will show that it also leads to a more efficient implementation.

⋆ The Netherlands Organisation for Scientific Research (NWO) is gratefully acknowledged for financial support.

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part IV, LNCS 3954, pp. 228–240, 2006.

This paper starts with a description of the "traditional" tensor voting method. We will redefine the operational definition of tensor voting in a neater way. Subsequently we will show that using steerable filter theory [3], we can create an implementation of tensor voting that consists of ordinary complex-valued convolutions. This is more efficient, since no algebraic calculations or interpolations are necessary anymore. We will evaluate the advantages of the new approach, show some examples, and finally we will draw conclusions.

2 Tensor Voting

2.1 Data Representation

In 2D tensor voting, local image features are encoded into a tensor field $\mathbf{H} : \Omega \rightarrow \mathbf{T}_2(\mathbb{R}^2)$, where $\Omega \subset \mathbb{R}^2$ is the image domain, and $\mathbf{T}_2(\mathbb{R}^2)$ denotes the set of symmetric, positive semidefinite tensors of tensor rank 2 (i.e., rank-2 tensors) on $\mathbb{R}^2$.

In the following, we shall denote the cartesian basis vectors in the image space by $\mathbf{e}_x$ and $\mathbf{e}_y$, respectively. Unless stated otherwise, all vectors and tensors will be expressed in this basis. In this basis, each tensor $\mathbf{A} \in \mathbf{T}_2(\mathbb{R}^2)$ can be written as a positive semidefinite symmetric 2×2 matrix. We call this the *matrix representation* of the tensor. We can decompose such matrix into its eigenvectors and eigenvalues

$$\mathbf{A} = \begin{pmatrix} a_{xx} & a_{xy} \\ a_{xy} & a_{yy} \end{pmatrix} = \lambda_1 \, \mathbf{e}_1 \, \mathbf{e}_1^{\mathrm{T}} + \lambda_2 \, \mathbf{e}_2 \, \mathbf{e}_2^{\mathrm{T}}, \tag{1}$$

where λ_1 and λ_2 are nonnegative eigenvalues ($\lambda_1 \geq \lambda_2 \geq 0$), and $\mathbf{e_1}$ and $\mathbf{e_2}$ are the orthonormal eigenvectors. A graphical illustration of such a tensor is an ellipse, see Figure 1a. In this representation, the following three properties become apparent

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

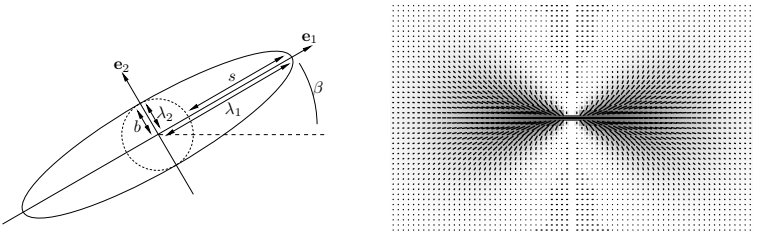
$$\text{Stickness} \quad s[\mathbf{A}] = \lambda_1 - \lambda_2 = \sqrt{\mathrm{tr}(\mathbf{A})^2 - 4 \, \det \mathbf{A}}, \tag{3}$$

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

Each tensor $\mathbf{A}$ is uniquely determined by these three scalars β (mod π, since the tensor has a 180° symmetry), $s \in \mathbb{R}^+ \cup \{0\}$, and $b \in \mathbb{R}^+ \cup \{0\}$. The stickness s is interpreted as a measure for the *orientation certainty* or a measure of anisotropy of the ellipse in orientation β. The ballness b is interpreted as a measure for the *orientation uncertainty* or isotropy.

There are two special cases for the positive semidefinite tensors: a *stick tensor* is a tensor with $b = 0$ and $s > 0$, and a *ball tensor* is an isotropic tensor, i.e. $s = 0$.

There are many ways to generate an input tensor field $\mathbf{H}$ from an input image. Medioni et al. assume that the input tensor field $\mathbf{H}$ is sparse, i.e. that most of the tensors in $\mathbf{H}$ are zero and therefore do not play any role. The way to generate a sparse tensor field (or in other words, a sparse set of tokens) out of an image is currently application-specific, but it is considered important to come to a more generic approach for generating tokens [4]. In this work, we assume that the obtained input field $\mathbf{H}$ is dense, to make



(a) (b)

Fig. 1. (a) Graphical representation of a rank-2 symmetric positive semidefinite tensor. (b) Example of a stick voting field. Gray scale indicates stickness value (darker mean higher value) and line segments indicate orientation.

the algorithms we develop generically applicable for both sparse and dense data, which is desirable as long as we do not have a well-justified method to sparsify our data.

As an example, a dense input tensor field could be obtained by creating feature images $\beta(\mathbf{x})$ and $s(\mathbf{x})$ by applying any type of orientation-selective filter with a $180°$ symmetry on the image data and taking the orientation with maximum filter response. The ballness $b(\mathbf{x})$ could be obtained by any isotropic filter on the image data. A tensor is uniquely determined by these three features, so we can now construct a tensor field $\mathbf{H}(\mathbf{x})$.

2.2 Voting Fields

TV uses a *stick tensor voting field* to incorporate the continuation of line structures. This voting field is a tensor field $\mathbf{V} : \Omega \rightarrow \mathbf{T}_2(\mathbb{R}^2)$, consisting entirely of stick tensors, in which the stickness of the tensors describes the likelihood that a feature at position $\mathbf{x}$ belongs to the *same* line structure as the feature positioned in the center $(0, 0)$ of the voting field with reference orientation $0°$. The orientation of the tensor at $\mathbf{x}$ describes the most probable orientation of a feature at that position. Rotated versions of $\mathbf{V}$ will be denoted by $\mathbf{V}^\alpha$, where α denotes the rotation angle

$$\mathbf{V}^\alpha(\mathbf{x}) = \mathbf{R}_\alpha \mathbf{V}(\mathbf{R}_\alpha^{-1}\mathbf{x})\mathbf{R}_\alpha^{-1}, \tag{5}$$

where

$$\mathbf{R}_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{6}$$

Medioni et al. [2] claim in their work that TV is model-free and that there is only one involved parameter, viz. scale. We, however, consider the voting field as the *model* used in tensor voting. Medioni's fundamental stick voting field is a model based on some assumptions on curves in images, but it is definitely not the only possible choice. One alternative voting field is discussed in Section 4. Figure 1b shows an example of a typical stick voting field.

2.3 Tensor Voting Operation

The idea of the TV operation is to let tensors *communicate* with each other by adding up contributions of neighboring tensors, resulting in a context-enhanced tensor field $\mathbf{U}$.

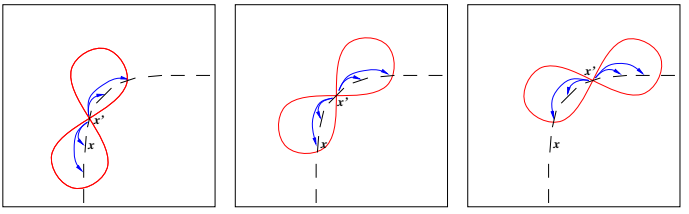


Fig. 2. Illustration of context communication within TV: the tensors communicate with each other using the stick voting field, which is indicated by the "8-shaped" contour. In this way the tensors strengthen each other.

Figure 2 illustrates the way the voting field is used on input data consisting of stick tensors. For each (nonzero) tensor $\mathbf{H}(\mathbf{x}')$, the voting field $\mathbf{V}^{\alpha}$ is centered at position $\mathbf{x}'$, aligned with the local orientation $\beta(\mathbf{H}(\mathbf{x}'))$: $\mathbf{V}^{\beta(\mathbf{H}(\mathbf{x}'))}(\mathbf{x}-\mathbf{x}')$. Then, to all tensors in a certain neighborhood (determined by the scale of the voting field), a weighted contribution (called a *vote*) $s(\mathbf{H}(\mathbf{x}'))\mathbf{V}^{\beta(\mathbf{H}(\mathbf{x}'))}(\mathbf{x}-\mathbf{x}')$ is added, where $\mathbf{x}$ is the position of the tensor that receives the vote. In other words, each tensor *broadcasts* contributions to the neighbors by appropriate alignment and rotation of the voting field. This results in the following operational definition for TV

$$\mathbf{U}(\mathbf{x}) = \int_{\Omega} s(\mathbf{H}(\mathbf{x}'))\, \mathbf{V}^{\beta(\mathbf{H}(\mathbf{x}'))}(\mathbf{x}-\mathbf{x}')\, d^2\mathbf{x}', \tag{7}$$

where the output is a tensor field $\mathbf{U}(\mathbf{x})$ with context-enhanced measures for orientation, stickness, and ballness. Note that in practice, the integral symbol in the previous equation amounts to a summation on a discrete grid.

Note that ballness b is not used in (7). The original TV formulation also incorporates *ball voting* [2], used to generate orientation estimates for β. Since we obtain estimates for β using local orientation-selective filters, we will not consider ball voting, hence the ballness b of all tensors in input tensor field $\mathbf{H}$ will be assumed to be zero.

2.4 Related Work

When using a dense input tensor field, the representation shows similarity with the well known *structure tensor*, and especially with the hour-glass smoothing filter extension described in [5], which shows resemblance with the tensor voting field. The difference is that the structure tensor is always constructed from the gradient of the image, while in tensor voting the input tensor field is considered a free choice. Also, the smoothing kernel used to smooth the structure tensor field is scalar-valued, while our voting field is tensorial. Tensor voting also shows resemblance with non-linear structure tensors [6], where anisotropic non-linear diffusion is applied on tensor images.

3 Steerable Tensor Voting

A technical difficulty in tensor voting is the alignment of the voting field with the orientation of a tensor, which needs to be done at every position. Since TV is a linear

operation (equation (7)), it is possible to handle these rotations in an efficient way. We will derive a method that we call *steerable tensor voting*. It is based on steerable filter theory as described by Freeman et al. [3]. We will first summarize this theory, and then explain steerable tensor voting.

3.1 Steerable Scalar Filters

One can rotate a scalar filter kernel $h(\mathbf{x})$ by α by counter rotating the filter domain: $h^\alpha(\mathbf{x}) = h(\mathbf{R}_\alpha^{-1}\mathbf{x})$. If we write the function in polar coordinates, denoted by $\tilde{h}(r,\phi)$, such that $\tilde{h}(r,\phi) = h(\mathbf{x})$ with $\mathbf{x} = (r\cos\phi, r\sin\phi)$, rotation becomes $\tilde{h}^\alpha(r,\phi) = \tilde{h}(r,\phi-\alpha)$.

Here we introduce the *spatial-angular Fourier decomposition* of a function $h : \mathbb{R}^2 \to \mathbb{C}$, which is given by

$$\tilde{h}_{m_s}(r) = \frac{1}{2\pi}\int_0^{2\pi} \tilde{h}(r,\phi)\, e^{-im_s\phi} d\phi, \tag{8}$$

and its inverse, the spatial composition, is

$$\tilde{h}(r,\phi) = \sum_{m_s\in\mathbb{Z}} \tilde{h}_{m_s}(r)\, e^{im_s\phi}. \tag{9}$$

A filter h is *steerable* if its spatial-angular Fourier composition (9) is a finite sum, i.e., there must be a finite number of nonzero Fourier coefficients $\tilde{h}_{m_s}(r)$. If a desired filter can be accurately approximated with a finite sum, we also call the filter steerable.

We write a steerable filter as $\tilde{h}(r,\phi) = \sum_{m_s=-M}^{M} f_{m_s}(r)e^{im_s\phi}$, where $M < \infty$ is the highest angular frequency of the filter kernel. Rotation of the steerable filter becomes

$$h^\alpha(\mathbf{x}) = \sum_{m_s=-M}^{M} \underbrace{e^{-im_s\alpha}}_{k_{m_s}(\alpha)} \underbrace{\tilde{h}_{m_s}(r)e^{im_s\phi}}_{h_{m_s}(\mathbf{x})}, \tag{10}$$

where $k_{m_s}(\alpha)$ are the linear coefficients as function of rotation angle α.

The filter response u^α of a filter h^α in orientation α is obtained by $u^\alpha(\mathbf{x}) = (f * h^\alpha)(\mathbf{x})$ where f is an image and "*" indicates convolution. If we substitute h^α by (10) in this equation and interchange sum and convolution, we get

$$u^\alpha(\mathbf{x}) = \sum_{m_s=-M}^{M} k_{m_s}(\alpha)\ (f * h_{m_s})(\mathbf{x}). \tag{11}$$

Hence, we can first convolve an image f with the $2M+1$ component functions h_{m_s} and then calculate the filter response for any orientation α, by simply taking the linear combination with coefficients $k_{m_s}(\alpha)$. This leads to an efficient method for oriented filtering if M is sufficiently small.

3.2 Rotation of 2-Tensors

A rank-2 symmetric tensor $\mathbf{A}$ can be rotated over an angle α as follows.

$$\mathbf{A}^\alpha = \mathbf{R}_\alpha \mathbf{A}\, \mathbf{R}_\alpha^{-1} \quad \text{with } \mathbf{R}_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{12}$$

It is more convenient to rewrite tensor $\mathbf{A}$ as a 3-tuple and to perform the tensor rotation with a single 3×3 matrix. That is, if we write $\overrightarrow{\mathbf{A}} = \begin{pmatrix} a_{xx} \\ a_{xy} \\ a_{yy} \end{pmatrix}$ then one can verify that

$$\overrightarrow{\mathbf{A}}^{\alpha} = \begin{pmatrix} \frac{1}{2}(1+\cos(2\,\alpha)) & -\sin(2\,\alpha) & \frac{1}{2}(1-\cos(2\,\alpha)) \\ \frac{1}{2}\sin(2\,\alpha) & \cos(2\,\alpha) & -\frac{1}{2}\sin(2\,\alpha) \\ \frac{1}{2}(1-\cos(2\,\alpha)) & \sin(2\,\alpha) & \frac{1}{2}(1+\cos(2\,\alpha)) \end{pmatrix} \overrightarrow{\mathbf{A}}. \tag{13}$$

It is a special property of the 2D rotation group that one can diagonalize the rotation matrix for all α by applying a similarity transformation $\mathbf{S}$

$$\overrightarrow{\mathbf{A}}^{\alpha} = \underbrace{\frac{1}{4}\begin{pmatrix} 2 & 1 & 1 \\ 0 & i & -i \\ 2 & -1 & -1 \end{pmatrix}}_{\mathbf{S}^{-1}} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & e^{-2i\,\alpha} & 0 \\ 0 & 0 & e^{2i\,\alpha} \end{pmatrix}}_{\mathbf{R}'_{\alpha}} \underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 1 & -2i & -1 \\ 1 & 2i & -1 \end{pmatrix}}_{\mathbf{S}} \overrightarrow{\mathbf{A}}. \tag{14}$$

So if we transform a tensor $\overrightarrow{\mathbf{A}}$ as

$$\overrightarrow{\mathbf{A}}' = \begin{pmatrix} A_0 \\ A_2 \\ A_{-2} \end{pmatrix} = \mathbf{S}\,\overrightarrow{\mathbf{A}} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & -2i & -1 \\ 1 & 2i & -1 \end{pmatrix} \begin{pmatrix} a_{xx} \\ a_{xy} \\ a_{yy} \end{pmatrix}, \tag{15}$$

we obtain components A_0, A_2, and A_{-2}, which are the tensor components in rotation-invariant subspaces. These components are rotated by a simple complex phase factor: $A^{\alpha}_{m_a} = e^{-im_a\alpha}A_{m_a}$ ($m_a = 0, -2, 2$), which directly follows from (14). Henceforth we will call these components the m_a-*components* of the tensor, and the transformation of (15) is called the *orientation-angular Fourier decomposition* of the tensor.

Note that the properties β, s, b defined in equations (2) to (4) can be easily described in terms of A_0, A_{-2} and A_2 using (15)

$$\beta = \frac{1}{2}\arg A_{-2} \qquad s = \sqrt{A_{-2}A_2} = |A_2| = |A_{-2}| \qquad b = \frac{1}{2}(A_0 - |A_2|). \tag{16}$$

Note also that $A_2 = \overline{A_{-2}}$ where $\overline{A_{-2}}$ is the complex conjugate of A_{-2}.

3.3 Steerable Tensor Filters

The concept of steerable filters can also be applied to tensor fields and to the voting field in particular, which rotates according to $\mathbf{V}^{\alpha}(\mathbf{x}) = \mathbf{R}_{\alpha}\mathbf{V}(\mathbf{R}_{\alpha}^{-1}\mathbf{x})\mathbf{R}_{\alpha}^{-1}$. In the previous subsection we showed how to decompose tensors in orientation-angular Fourier components and how to rotate them. For the voting field we get $V^{\alpha}_{m_a}(\mathbf{x}) = e^{-im_a\alpha}V_{m_a}(\mathbf{R}_{\alpha}^{-1}\mathbf{x})$ for $m_a = -2, 0, 2$. These three V_{m_a} functions are of the form $\mathbb{R}^2 \to \mathbb{C}$ and can be made steerable in the same way as scalar filters. So, a voting field is steerable if for all m_a-components ($m_a = 0, -2, 2$) of the tensor field we can write $V_{m_a}(\mathbf{x}) = \sum_{m_s=-M}^{M} \tilde{V}_{m_a m_s}(r)e^{im_s\phi}$. Rotation becomes

$$V^{\alpha}_{m_a}(\mathbf{x}) = \sum_{m_s=-M}^{M} e^{-i(m_a+m_s)\alpha} \underbrace{\tilde{V}_{m_a m_s}(r)\, e^{im_s\phi}}_{V_{m_a m_s}(\mathbf{x})}, \tag{17}$$

where $V_{m_a m_s}(\mathbf{x})$ are the basis filters and $e^{-i(m_a+m_s)\alpha}$ are the linear coefficient functions of rotation angle α. Filling the previous equation into (7), and writing $\mathbf{U}$ in its m_a-components according to (15) results in

$$
\begin{aligned}
U_{m_a}(\mathbf{x}) &= \int_{\Omega} s(\mathbf{H}(\mathbf{x}'))\, V_{m_a}^{\beta(\mathbf{H}(\mathbf{x}'))}(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \\
&= \int_{\Omega} s(\mathbf{H}(\mathbf{x}')) \left(\sum_{m_s=-M}^{M} e^{-i(m_a+m_s)\beta(\mathbf{H}(\mathbf{x}'))}\, V_{m_a m_s}(\mathbf{x}-\mathbf{x}') \right) d\mathbf{x}' \\
&= \sum_{m_s=-M}^{M} \int_{\Omega} \left(s(\mathbf{H}(\mathbf{x}'))\, e^{-i(m_a+m_s)\beta(\mathbf{H}(\mathbf{x}'))} \right) V_{m_a m_s}(\mathbf{x}-\mathbf{x}')d\mathbf{x}' \\
&= \sum_{m_s=-M}^{M} \left(\left(s(\mathbf{H})\, e^{-i(m_a+m_s)\beta(\mathbf{H})} \right) * V_{m_a m_s} \right)(\mathbf{x}). \qquad (18)
\end{aligned}
$$

This important result states that, as opposed to non-steerable TV, we can apply TV simply by calculating $2 \cdot (M+1)$ convolutions, viz. for each m_a component we need $M+1$ m_s-terms, since all odd m_s components are zero for $180°$-symmetric voting fields. Furthermore, taking into account that $U_2(\mathbf{x}) = \overline{U_{-2}(\mathbf{x})}$ we see that we only have to calculate and $U_0(\mathbf{x})$ and $U_2(\mathbf{x})$. Notice also that the convolutions involve relatively large kernels, meaning that they can possibly be done more efficiently via the Fourier domain, i.e. $A * B = \mathcal{F}^{-1}[\mathcal{F}[A] \cdot \mathcal{F}[B]]$, where $\mathcal{F}$ denotes the spatial Fourier transform.

4 Voting Fields

The stick voting field can be freely chosen in tensor voting. In this section we will treat two different voting fields and state some qualitative differences between them.

4.1 Medioni's Voting Field

Medioni et al. [2] assume that the best connection between two points with one orientation imposed is a circular arc. If one point is horizontally oriented and the angle of the vector connecting the two points is ϕ, then the angle at the other point is 2ϕ. This *cocircularity model* is encoded in a tensor field consisting of stick tensors with $\lambda_1 = 1$ (and $\lambda_2 = 0$) as $\mathbf{c}\mathbf{c}^{\mathrm{T}}$ with $\mathbf{c} = \begin{pmatrix} \cos 2\phi \\ \sin 2\phi \end{pmatrix}$ (cf. (1)).

To obtain a locally confined voting field the cocircularity pattern is modulated with a function that decays with radius curve length and curvature. This yields the following voting field

$$
\tilde{\mathbf{V}}(r,\phi) = e^{-\left(\frac{\phi r}{\sigma_{\text{ctx}} \sin\phi}\right)^2 - p\left(\frac{2\sigma_{\text{ctx}} \sin\phi}{r}\right)^2} \begin{pmatrix} 1+\cos(4\phi) & \sin(4\phi) \\ \sin(4\phi) & 1-\cos(4\phi) \end{pmatrix} \qquad (19)
$$

where σ_{ctx} is the scale of the voting field, p is a dimensionless constant describing the relative weight of the curvature. In practice, points above an below the diagonals $\phi = \pm\pi/4 \mod \pi$ in the field are considered too unlikely to belong to the same structure as the point in the center of the field, so the field is truncated for these values of ϕ.

There are two drawbacks of this voting field concerning steerable tensor voting. First, there is no simple analytic expression for the components $V_{m_a m_s}(\mathbf{x})$, so one should calculate the steerable components of this kernel numerically. Second, the field has an infinite number of m_s-components, so we have to cut the sum over m_s such that we get a reasonable approximation. Therefore, in the next subsection we propose a different voting field, which is more suitable for steerable tensor voting.

4.2 Bandlimited Voting Field

Here we propose a bandlimited voting field, which is especially useful for steerable tensor voting, since it has a limited number of spatial-angular Fourier components. The decay function is similar to the one for instance used in [7].

Similar to Medioni's voting field, we assume that the best connection between two points with one orientation imposed is a circular arc. Now we modulate the cocircularity pattern with a function that decays with radius r. We choose a Gaussian decay as function of r. To penalize high-curvature arcs, we must also account for some periodic modulation that reaches its maximum at $\phi = 0 \mod \pi$ and minimum for $\phi = \pi/2 \mod \pi$. This is achieved 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. The voting field now becomes, expressed in spatial polar coordinates

$$\tilde{\mathbf{V}}(r,\phi) = \frac{1}{G}\, e^{\frac{-r^2}{2\sigma_{\text{ctx}}^2}} \cos^{2n}(\phi) \begin{pmatrix} 1+\cos(4\phi) & \sin(4\phi) \\ \sin(4\phi) & 1-\cos(4\phi) \end{pmatrix}, \tag{20}$$

where $\sigma_{\text{ctx}} \in \mathbb{R}^+$ is the scale of the voting field. The factor G is a normalization factor. This voting field is depicted in Figure 1b. In the following, to get simpler equations, we will use $G = 1/16$ and $n = 2$.

We apply orientation-angular (15) and spatial-angular (8) Fourier decomposition to this voting field. The spatial-angular Fourier decomposition is trivial if we first replace all trigonometric functions by exponentials and expand these exponentials.

$$\begin{pmatrix} \tilde{V}_0(r,\phi) \\ \tilde{V}_2(r,\phi) \\ \tilde{V}_{-2}(r,\phi) \end{pmatrix} = e^{-\frac{r^2}{2\sigma_{\text{ctx}}^2}} \begin{pmatrix} e^{-i4\phi}+4e^{-i2\phi}+6+4e^{i2\phi}+e^{i4\phi} \\ e^{-i8\phi}+4e^{-i6\phi}+6e^{-i4\phi}+4e^{-i2\phi}+1 \\ 1+4e^{i2\phi}+6e^{i4\phi}+4e^{i6\phi}+e^{i8\phi} \end{pmatrix} \tag{21}$$

For every m_a-component we have effectively 5 m_s-components. We can now write this filter in the steerable form cf. (17)

$$\begin{pmatrix} V_0^\alpha(\mathbf{x}) \\ V_2^\alpha(\mathbf{x}) \\ V_{-2}^\alpha(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} 0 & 0 & e^{4i\alpha} & 4e^{2i\alpha} & 6 & 4e^{-2i\alpha} & e^{-4i\alpha} & 0 & 0 \\ e^{6i\alpha} & 4e^{4i\alpha} & 6e^{2i\alpha} & 4 & e^{-2i\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e^{2i\alpha} & 4 & 6e^{-2i\alpha} & 4e^{-4i\alpha} & e^{-6i\alpha} \end{pmatrix} \begin{pmatrix} w_{-8}(\mathbf{x}) \\ w_{-6}(\mathbf{x}) \\ \vdots \\ w_6(\mathbf{x}) \\ w_8(\mathbf{x}) \end{pmatrix} \tag{22}$$

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 defined by $\tilde{w}_{m_s}(r,\phi) = e^{-\frac{r^2}{2\sigma_{\text{ctx}}^2}} e^{im_s\phi}$. In cartesian coordinates they are given by

$$w_{m_s}(\mathbf{x}) = e^{-\frac{x^2+y^2}{2\sigma_{\text{ctx}}^2}} \left(\frac{x+iy}{\sqrt{x^2+y^2}}\right)^{m_s}, \quad \text{for } \mathbf{x} \neq (0,0). \tag{23}$$

Using (18) we can implement steerable tensor voting for this voting field, as follows. The filter kernels $w_{m_s}(\mathbf{x})$ are tabulated for $m_s = 0, 2, 4, 6, 8$ (note that $w_{-m_s} = \overline{w_{m_s}}$). Given the stickness s and orientation β of tensors in $\mathbf{H}$, we need to calculate a number of complex-valued feature images $c_{m_s}(\mathbf{x}) = s(\mathbf{H}(\mathbf{x}))\; e^{-im_s\beta(\mathbf{H}(\mathbf{x}))}$ for $m_s = 0, 2, 4, 6$. Now, we can calculate the resulting $m_a = -2$ part by

$$U_{-2}(\mathbf{x}) = (w_0 * \overline{c_2}) + 4(w_2 * c_0) + 6(w_4 * c_2) + 4(w_6 * c_4) + (w_8 * c_6), \tag{24}$$

where $\overline{c_2}$ denotes complex conjugate and $\overline{c_2} = c_{-2}$. The $m_a = 2$ part does not need to be calculated explicitly, because it is simply the complex conjugate of the $m_a = -2$ part. The $m_a = 0$ part can be calculated by

$$\begin{aligned} U_0(\mathbf{x}) &= (\overline{w_4} * \overline{c_4}) + 4(\overline{w_2} * \overline{c_2}) + 6(w_0 * c_0) + 4(w_2 * c_2) + (w_4 * c_4) \\ &= \mathrm{Re}\big(6(w_0 * c_0) + 8(w_2 * c_2) + 2(w_4 * c_4)\big). \end{aligned} \tag{25}$$

So TV with this voting field requires 8 convolutions. In the resulting context-enhanced tensor field $\mathbf{U}$, we are interested in the orientation, stickness, and ballness. These measures are calculated using (16).

5 Computational Efficiency

In this section we compare three different versions of tensor voting:

- *Normal TV*: "Normal" tensor voting where the stick voting field is calculated algebraically at every position;
- *Steerable TV spatial*: Steerable tensor voting using spatial convolutions;
- *Steerable TV FFT*: Steerable tensor voting using FFT.

5.1 Computational Complexity

A typical TV implementation scans through the entire image, and collects or broadcasts information from or to the neighborhood of every tensor. If our image is square with a size of $s \times s$ pixels and the voting field kernel has a size $k \times k$, then s^2k^2 tensor additions need to take place. The order of this algorithm is thus $\mathcal{O}(s^2k^2)$. Steerable tensor voting consists of a sum of complex-valued 2D convolutions. If they are implemented in the spatial domain, the order is $\mathcal{O}(s^2k^2)$ as well. However, if the convolutions are implemented through the Fourier domain using a 2D FFT implementation, the order is reduced to $\mathcal{O}(s^2 \log s)$.

5.2 Speed Comparison

To give an impression of the differences in speed we did some speed measurements. All algorithms were implemented in C++, and compiled using Microsoft Visual C++. To make the comparison fair, all 3 variants use the bandlimited voting field of Subsection 4.2, implying that they all give the same results. We use the FFTW library for the FFT (see http://www.fftw.org/). All numbers are stored using data type "double". For normal TV and steerable TV spatial the kernel has a pixel size of $\frac{1}{4}$ times the size of the image, i.e. the scale of the kernel scales proportionally to the image size. The steerable TV implementation uses a voting field with 9 steerable components. We use random dense stickness and orientation maps as input, since the speed of the algorithms is not dependent on the contents of the input data if the input data is dense.

Figure 3 shows computation times as function of image size, measured on an AMD Athlon 64 X2 4400+ running on Windows XP at 2.3 GHz. It is clear that steerable TV FFT performs fastest, followed by steerable TV spatial. The normal TV version is much slower. As example to show the large differences, on a 512×512 image, steerable TV

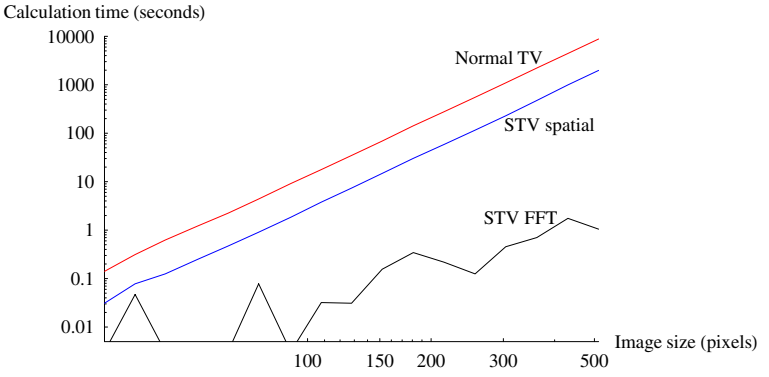


Fig. 3. Speed measurements of three different TV algorithms as function of image size. STV = Steerable tensor voting. See Subsection 5.2 for details.

FFT takes 1.05 seconds, while steerable TV spatial takes 1980 seconds, and normal TV takes 8803 seconds. Since the graph is a log-log plot, the slope of the curves indicates the computational complexity of the algorithm. As expected, steerable TV with FFT has a smaller slope. The latter curve shows an irregular trajectory, due to some technicalities of the FFT algorithm. The FFT implementation is more efficient if the data size is a power of 2 a product of small prime factors. The normal TV implementation is slower because of the analytic calculations that are required for every vote[1].

Convolutions and FFT's only involve multiplications, additions, and memory accesses in a very regular fashion. These operations are therefore very suitable for efficient implementations exploiting cache memory and parallelism. A graphical processing unit (GPU) might be suitable to host an extremely fast implementation of steerable tensor voting [8].

As final remark, note that if the tensor field is very sparse, steerable TV may perform worse since it does not benefit from the possibility to skip zero-valued tensors during the voting process.

6 Examples of 2D Steerable Tensor Voting

Steerable tensor voting is a new computational method for tensor voting, which yields the same result as normal tensor voting if exactly the same voting field is used. Therefore, in this section we do not perform comparison of results of the two methods, but only show two applications that require computationally efficient algorithms.

In the examples, our approach for the curve extraction process is as follows. The input stickness s and the orientation β are constructed using first order (for edges) or second order (for ridges) Gaussian derivatives. To enhance these data, a steerable tensor voting step is performed according to (18). Spurious responses caused by noise in the image cause that the resulting stickness image is not sufficiently enhanced: for instance,

[1] Alternatively, one could also precalculate the voting field in one or a limited number of orientations and then interpolate. However, this will lead to discretization errors and irregular memory accesses, and therefore possible caching problems on typical computer systems.

the resulting curves might still have gaps. To get more consistent curves, 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 on the thinned image. So, the first TV step is on dense data (prior to any hard decision step), the second step is on sparse data.

6.1 Electrophysiology Catheters

An interesting example application is the detection of Electrophysiology (EP) catheters in X-ray fluoroscopy images. These images are generated during heart catheterization procedures. Figure 4 shows an example of such an image, the result obtained with steerable tensor voting and the final result after an EP-catheter specific extraction algorithm is applied on the resulting stickness image. We did an extensive evaluation on this medical application, and clearly our extraction performance increased using tensor voting compared to not using tensor voting. More details can be found in [9].

6.2 Ultrasound Kidney

Figure 5 shows the results on the ultrasound image of a kidney. The line segment extraction in subimages (c) en (f) is achieved by applying thinning, and extraction of strings of

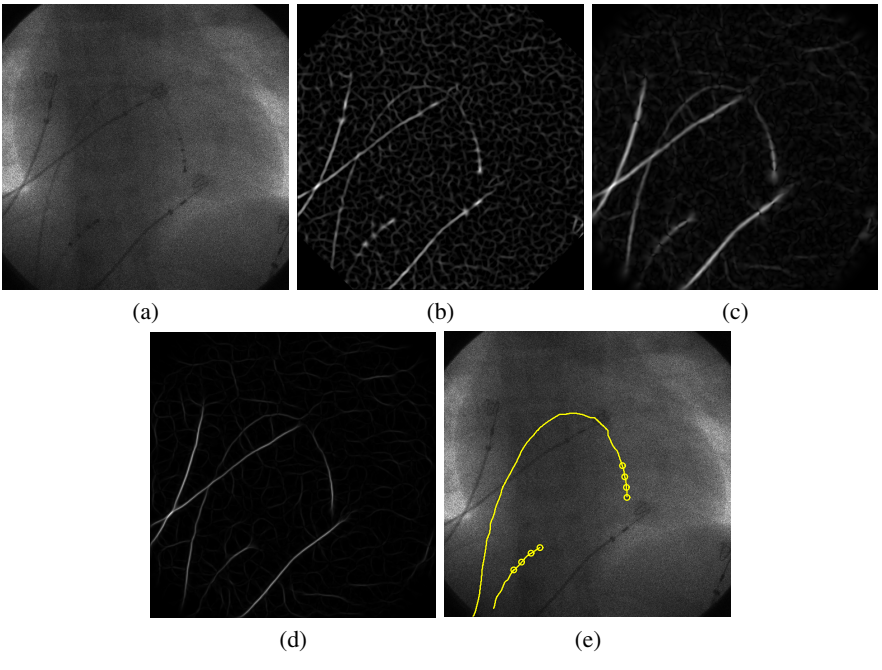


Fig. 4. Electrophysiology catheter extraction example. (b) Original noisy image, used as input for this example. Size 512×512 pixels. (b) Local ridgeness image (i.e. the largest eigenvalue of Hessian constructed using 2nd order Gaussian derivatives with scale $\sigma = 3.4$ pixels). (c) Result of first tensor voting step with $\sigma_{\text{ctx}} = 15$. (d) Result of a second tensor voting step with $\sigma_{\text{ctx}} = 7.5$. (e) Final extraction result using EP-catheter specific extraction algorithm.

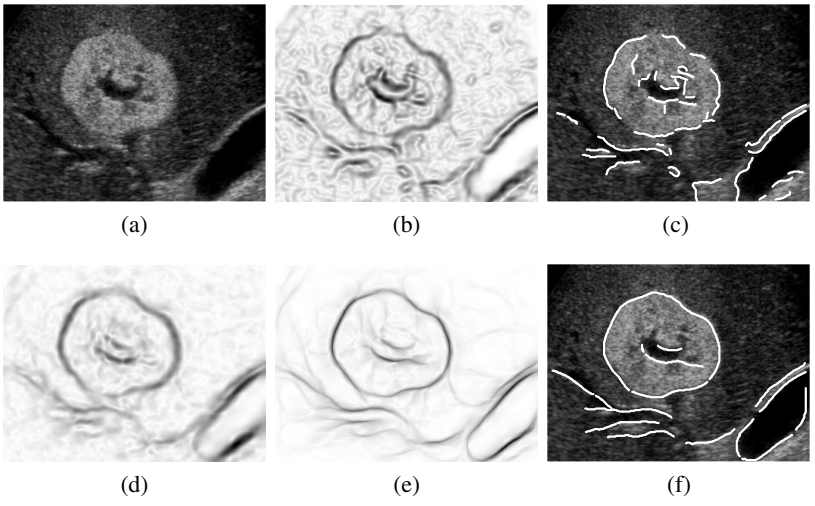


Fig. 5. Example of tensor voting on an ultrasound image of a kidney. (a) Original image, size 345×260 pixels. (b) Gradient magnitude with $\sigma = 3$ pixels. (c) Extracted line segments using (b). (d) Result after first tensor voting step with $\sigma_{\text{ctx}} = 15$. (e) Result after second tensor voting step with same settings. (f) Extracted line segments using image (e). In both (c) and (f), 1250 pixels were extracted for the sake of comparison.

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.

7 Conclusion

The main conclusion of this paper is that tensor voting can be made steerable. We are able to write tensor voting as a summation of a number of complex-valued convolutions. We showed that two-dimensional steerable tensor voting is computationally more efficient on dense tensor fields. The highest speed gain is achieved if we implement steerable tensor voting using FFT. A GPU implementation might lead to an even faster implementation.

Another point we made is that the voting field of Medioni is not the only possible voting field. We proposed the bandlimited voting field as alternative. The voting field could also be made more application-specific by gathering statistics on curves in a specific application. Or it can be made as generic as possible by only using the random walker prior, leading to the stochastic completion field [10, 11].

Our examples show that the method is especially feasible for applications where thin noisy line-structures must be detected. These kind of problems often arise in the field of medical image analysis.

Tensor voting and related methods are promising methods for robust extraction of line-like structures in images. There are still a lot of challenges to be faced, such as 3D steerable tensor voting and multi-scale tensor voting.

References

1. Guy, G., Medioni, G.: Inferring global perceptual contours from local features. International Journal of Computer Vision **20**(1–2) (1996) 113–33
2. Medioni, G., Lee, M.S., Tang, C.K.: A Computational Framework for Segmentation and Grouping. Elsevier (2000)
3. Freeman, W.T., Adelson, E.H.: The design and use of steerable filters. IEEE Trans. Pattern Analysis and Machine Intelligence **13**(9) (1991) 891–906
4. Medioni, G., Mordohai, P.: The Tensor Voting Framework. IMSC Press Multimedia Series. In: Emerging Topics in Computer Vision. Prentice Hall (2004) 191–252
5. Köthe, U.: Edge and junction detection with an improved structure tensor. In Michaelis, B., Krell, G., eds.: Pattern Recognition, Proc. of 25th DAGM Symposium, Magdeburg 2003. Volume 2781 of Lecture Notes in Computer Science., Heidelberg, Springer (2003) 25–32
6. Brox, T., Weickert, J., Burgeth, B., Mrázek, P.: Nonlinear structure tensors. Image and Vision Computing **24**(1) (2006) 41–55
7. Heitger, F., von der Heydt, R.: A computational model of neural contour processing. In: Proc. 4th Int. Conf. Computer Vision, Washington D.C.: IEEE Computer Society Press (1993) 32–40
8. Moreland, K., Angel, E.: The FFT on a GPU. In: SIGGRAPH/Eurographics Workshop on Graphics Hardware. (2003) 112–119
9. Franken, E., van Almsick, M., Rongen, P., ter Haar Romeny, B.: Context-enhanced detection of electrophysiology catheters in X-ray fluoroscopy images. Conference Poster, European Conference of Radiology (ECR) (2005) http://www.ecr.org/.
10. Williams, L.R., Jacobs, D.W.: Stochastic completion fields: a neural model of illusory contour shape and salience. Neural Comput. **9**(4) (1997) 837–858
11. Almsick, M.A., Duits, R., Franken, E., ter Haar Romeny, B.: From stochastic completion fields to tensor voting. In Fogh Olsen, O., Florack, L., Kuijper, A., eds.: Deep Structure Singularities and Computer Vision. Volume 3753 of Lecture Notes in Computer Science., Springer-Verlag (2005) 124–134