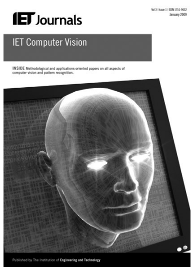
Published in IET Computer Vision
Received on 17th April 2014
Revised on 8th June 2014
Accepted on 8th July 2014
doi: 10.1049/iet-cvi.2014.0103



ISSN 1751-9632

Perceptual grouping by tensor voting: a comparative survey of recent approaches

Emmanuel Maggiori[1,2], *Hugo Luis Manterola*[2,3], *Mariana del Fresno*[2,4]
[1]AYIN and STARS, Inria Sophia Antipolis F-06902, France
[2]PLADEMA, Fac. de Cs. Exactas, UNCBPA, Pinto 399, 7000 Tandil, Argentina
[3]Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina
[4]Comisión de Investigaciones Científicas, Provincia de Buenos Aires (CIC), Argentina
E-mail: emmanuel.maggiori@inria.fr

Abstract: Tensor voting is a computational framework that addresses the problem of perceptual organisation. It was designed to convey human perception principles into a unified framework that can be adapted to extract visually salient elements from possibly noisy or corrupted images. The original formulation featured some concerns that made it difficult or impractical to be applied directly. Therefore, several partial or total theoretical reformulations or augmentations have been proposed. These almost parallel publication were presented in different directions, with different priorities and even in a different notation. Thus, the authors observed the need for a coherent description and comparison of the different proposals. This work, after describing the original approach of tensor voting, reviews and explains a number of high impact theoretical modifications in a self-contained manner and including possible future directions of work. The authors have selected and organised a number of formulations and unified the way the problem is addressed across the different proposals. The aim of this study is to contribute with a modern comprehensive source of information on the theoretical aspects of tensor voting.

1 Introduction

Perceptual grouping is the human vision ability to extract significant features from a scene without prior knowledge of the domain. If no such knowledge is required, a set of laws must then rule the perceptual system. These laws have been the object of study of Gestalt psychology [1].

Many computer vision problems, however, heterogeneous, can be posed in terms of perceptual grouping. Tensor voting is inspired on human vision and provides a framework to extract perceptually salient features from clouds of points. Although generalised to an *n*-dimensional space with successful application results, it was originally used to obtain continuous smooth curves and meshes in two-dimensional (2D) and three-dimensional (3D) spaces, respectively. It is aimed at obtaining visually salient elements despite the existence of gaps in the features or noise. Tensor voting might constitute a good choice when there is a need for inferring data based on good continuation and proximity constraints. For instance, in [2], the authors have recently pointed out the difficulty in processing electron tomography images because of noise, low contrast and distortion, and dealt with these issues by using the tensor voting framework in 3D in order to detect membranes in the images.

Tensor voting is different to other methods of feature inference because it is not required to explicitly define an objective function and optimise it. Input tokens, which are encoded as ‘tensors’, communicate locally with their neighbours by casting ‘votes’. The design of the voting fields, which are accumulated at every location, is where Gestalt principles are introduced. The goal is that tokens that are mutually compatible reinforce each other, while the value of irrelevant tokens is disregarded.

Even though the technique was initially presented as a way of fitting surfaces to clouds of points, it has been widely applied to extract information from images.

Most literature on the original formulation of tensor voting was published in its pioneering papers [3, 4] later followed by a book on the subject that constitutes a comprehensive source of information on the technique and its potential applications [5]. A synthetic presentation of the method including some advances was later published as a chapter in a book that describes emerging topics in computer vision [6].

The original formulation offered promising results. However, it featured some key concerns. For example, the execution time was prohibitive because of the need for numerical integration. Another limitation was that it could not handle discontinuities in the detected features, so there was no way to treat the endpoints of an extracted element.

A few points were addressed soon after the original formulation. However, other concerns were dealt with over a decade after the publication of the original literature on tensor voting, in a sort of revival of the theoretical research on the matter.

The aim of this paper is to review the different recent theoretical approaches to the tensor voting technique, including those that constitute a relevant modification or augmentation to the original formulation.

The focus of this paper is to present highly relevant theoretical advances, which we study in a self-contained manner. This work is then aimed at constituting a comprehensive source of information on past, present and future theoretical directions of the tensor voting framework.

This paper is organised as follows: firstly, in Section 2, a broad explanation of the technique is included, with special care to provide the reader with perceptual interpretations of the mathematical background. In Section 3, the difficulties reported throughout the literature concerning the original formulation are reviewed to later describe reformulations of the technique in Section 4. In this section, observations are done separately for each of the different proposals and future directions of research are suggested. In addition, a theoretical comparison among the different techniques is included in this section, in order to synthetically show the virtues of every formulation and the trade-off points. In Section 5, relevant augmentations to the framework that extend its scope and capabilities are described. The way different computer vision problems have been adapted to be solved in a tensor voting context are later reviewed in Section 6. Finally, conclusions are presented in Section 7.

2 Original tensor voting

In this section, the original tensor voting (OTV) formulation [3–5] is reviewed. The general outline of the technique is initially presented without mathematical rigour to show the procedure from a high-level perspective. We later describe the framework in detail with a special emphasis on the intuitive interpretation of the procedure. In this second part, we include the motivation to use tensors and the way they are handled to represent data, followed by a description of the design of the fields that are needed during the voting steps. The ways to extract features after the voting are then described, followed by an explanation of the generalisation of the framework to an n-dimensional space.

2.1 Outline of the technique

The general outline of the tensor voting technique is depicted in Fig. 1.

Firstly, the input is encoded as a set of tensor tokens. Given that tensor voting is a framework, the meaning of the input tokens is not specified and might vary depending on the problem. As a general rule, unoriented tensors should represent the locations at which a feature is supposed to exist, and oriented tensors should be used to indicate the direction of the feature at that location if it is known (in general the direction of the normal line, plane or hyperplane is used).

Therefore, prior to using the technique, a decision on how to encode the data in tensors has to be made. For example, if the technique will be used to detect edges in images, tensors representing normals can be used [7].

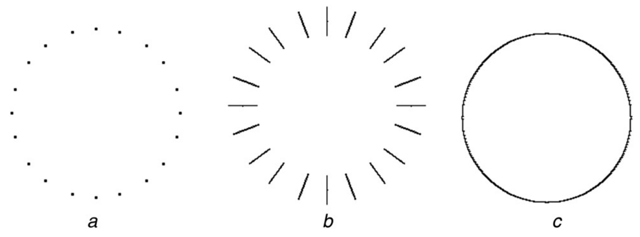


Fig. 2 *Tokens feature the preferred orientations according to the other tokens*

a Sparse unoriented input tokens
b Result after sparse voting
c Feature inference after dense voting

The general rule is that the input tokens should be ‘perfect tensors’ [8]. This means that the tokens should encode a certain ‘pure’ feature and not a combination of those. For example, a normal should be encoded as a stick-shaped tensor and a point as a ball-shaped tensor. Formally, perfect tensors are those whose non-zero eigenvalues are all equal.

After encoding the input in perfect tensors, different voting procedures take place. These involve the propagation of the information they encode to their neighbourhood.

The first voting process is the so-called ‘sparse’ voting in which tensors communicate only with each other. Every tensor collects at its location the votes cast by the others. These votes are the continuations that every tensor proposes for the feature they encode. For example, an oriented tensor will cast the most prominent votes at locations along the tangent of the curve it encodes, because this is the most likely continuation of a smooth curve. When the sparse voting is complete, every token contains a refined tensor. In the case unoriented tokens were present, after the sparse voting the normals are estimated and now the tokens feature the preferred orientations according to the other tokens. This is illustrated in Fig. 2 where Fig. 2*a* shows a set of unoriented input tokens in 2D and Fig. 2*b* shows the new refined tensors after the sparse voting. The normals to an underlying circumference have been successfully inferred without the need of indicating which feature is being sought.

The following step, after refining the original input, is to obtain a new set of perfect tensors. This is done by discarding components other than the most salient one [6]. For example, if there are non-stick tensors after the voting, only the stick part of it is forwarded to the next step. In other words, the uncertainty is ignored. The result is a corrected version of the input.

A second voting is then carried out. This time votes are collected at every location in space, even where no input tokens were present. This stage is known as the ‘dense’ voting. The result is a tensor field which can be decomposed and evaluated to obtain the underlying features that were revealed. Fig. 2*c* shows the result of a feature extraction process after a dense voting with the tokens of Fig. 2*a* which were refined as in Fig. 2*b*. A whole continuous circumference has been inferred.

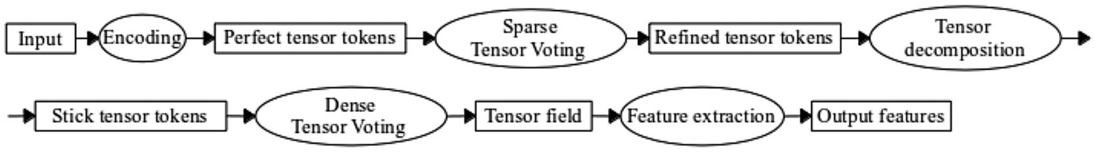


Fig. 1 *Outline of the tensor voting technique*

2.2 Encoding data into tensors

Second-order symmetric tensors are selected to encode the data of the underlying structures in the input image. In 2D, these tensors can be visualised as ellipses. The greater axis of the ellipse represents the orientation of the tensor, usually used to encode the normal direction to a curve. Tangent directions could be equally encoded. However, in [6], the authors suggest that in 3D and higher dimensions the use of normals and not tangents, although still equivalent, is more convenient so that intersections can be obtained simply by computing the union of normal spaces of each of the intersecting structures by using a Gram–Schmidt procedure [9].

The second axis of the ellipse, orthogonal to the greater one, encodes the uncertainty about the main orientation: the thinner the ellipse, the higher the confidence of the direction. The thinnest possible tensor, the so-called 'stick' tensor, is an ellipse with zero width. The tensor whose axes are equal in length is the 'ball' tensor, representing the total uncertainty of the direction since their components are totally contradictory.

At this point, it is important to remark that the main motivation is to use second-order symmetric tensors and not simply first-order vectors. By using a vector, the certainty of the represented direction can be encoded in its length. However, the uncertainty of its direction cannot be measured.

In the tensorial representation, even if the direction is totally unknown and the tensor is a ball, the size of the ball can be used as a measure of the degree of uncertainty. This way, a high radius ball represents the junction or intersection of various features so that the direction at that point cannot be determined. That is, it can be stated with high confidence that the direction is undetermined. A low radius ball represents noise or the lack of information, since there is no direction, nor evidence of ambiguity in the direction. In other words, it is not the same to state that no information of direction is present as to state that the direction is likely to be undetermined. This concept is well captured in the tensorial representation and a key part of the tensor voting framework, allowing preservation of junctions and intersections among the features detected.

In 2D, the tensors might take two distinct extremal shapes, a 'ball' and a 'stick', as well as a combination of both. This is shown in Fig. 3. In Fig. 4, the addition operation between tensors is illustrated. The case in which orthogonal tensors are added (which is done when votes are collected) results in a high radius ball, representing that the direction is uncertain.

A second-order symmetric tensor T in 2D can be represented as a non-negative definite 2×2 symmetric matrix, which can be generated by the following equation

$$T = \lambda_1 \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \lambda_2 \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T \tag{1}$$

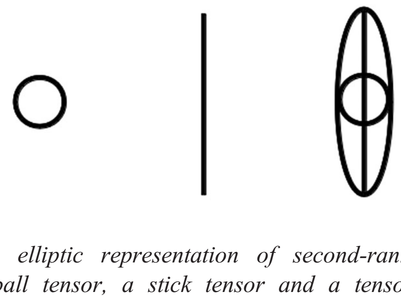


Fig. 3 *2D elliptic representation of second-rank symmetric tensors: a ball tensor, a stick tensor and a tensor with both components*

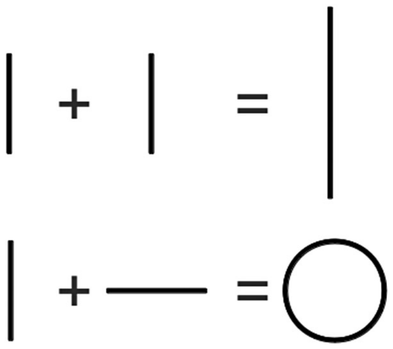


Fig. 4 *Tensors encode the preferred orientations according to others and facilitate feature inference*

where $\hat{\boldsymbol{e}}_1$ and $\hat{\boldsymbol{e}}_2$ are the eigenvectors of T and λ_1 and λ_2 their corresponding eigenvalues (being $\lambda_1 \geq \lambda_2$). The outer product operation $\hat{\boldsymbol{e}}_i \hat{\boldsymbol{e}}_i^T$ is frequently used throughout the literature to generate a stick tensor in the direction of $\hat{\boldsymbol{e}}_i$.

In practice, these tensors can be represented with the upper triangle of the matrix (since it is symmetric) or in a decomposed form, with the coordinates or the angle of the main eigenvector and the two eigenvalues.

Equation (1) can be conveniently rewritten as in the following equation

$$T = (\lambda_1 - \lambda_2) \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \lambda_2 (\hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T) \tag{2}$$

$\lambda_1 - \lambda_2$ is the so-called 'saliency' of the tensor, a measure of the confidence of its encoded direction. This is intuitive since it computes the difference between its contradictory eigenvalues. λ_2 is the so-called 'junctionness' of the tensor, a measure of the level of contradiction in the direction of the feature encoded. This is also natural observing that in the ellipse representation λ_2 is related to the size of the minor axis, which contradicts the main direction of the ellipse.

In 3D, an ellipsoid can be used to visualise the tensor. A stick tensor in 3D represents with high confidence the existence of a surface. A ball tensor represents the total uncertainty of the direction, whose radius is again used as a measure of confidence of the uncertainty. An additional extreme tensor exists in 3D, the so-called 'plate tensor', which can be naturally visualised as a plate in space. This tensor encodes an underlying curve. These concepts are illustrated in Fig. 5.

3D tensors can be expressed as in the following equation

$$T = \lambda_1 \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \lambda_2 \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T + \lambda_3 \hat{\boldsymbol{e}}_3 \hat{\boldsymbol{e}}_3^T \tag{3}$$

with three eigenvectors $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$, and their corresponding eigenvalues λ_1, λ_2 and λ_3 (with $\lambda_1 \geq \lambda_2 \geq \lambda_3$). As in 2D, a decomposed form is convenient to reveal the properties of the element encoded by the tensor

$$\begin{aligned} T = (\lambda_1 - \lambda_2) \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T (\lambda_2 - \lambda_3) \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T \\ + \lambda_3 (\hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T + \hat{\boldsymbol{e}}_3 \hat{\boldsymbol{e}}_3^T) \end{aligned} \tag{4}$$

In this case, $\lambda_1 - \lambda_2$ is the 'surfaceness' of the tensor: the higher the difference between λ_1 and λ_2, the more the tensor resembles a stick (with the direction of $\boldsymbol{e}_1$). $\lambda_2 - \lambda_3$ represents the 'curveness': the higher the difference between those, the more the tensor resembles a plate tensor (with the

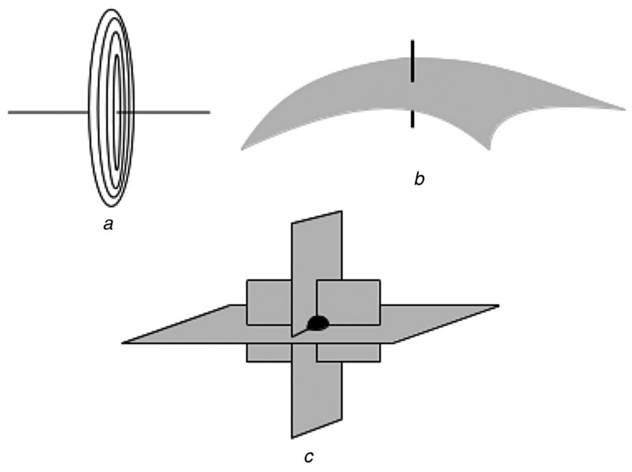


Fig. 5 *In 3D*

a Plate is used to encode the normal plane to a curve
b Stick: the normal direction to a surface
c Ball: the existence of an intersection

direction of $\hat{\boldsymbol{e}}_2$). Finally, λ_3, usually called the 'pointness' in the literature, measures the uncertainty of the direction, analogously to the lower eigenvalue in the 2D case.

In Fig. 6, the components of a 3D tensor are illustrated.

2.3 Fundamental stick voting field

During the voting, tensors cast votes in different positions along the space. These votes are also tensors and encode information in the same way and with the same meaning. The votes at every location constitute the voting field of a tensor. The stick voting field, that is, the field produced by a stick tensor, is fundamental in the sense that every other field can be computed after it.

The design of the stick voting field is crucial in the framework, since it is at this point that human perception principles are introduced. The votes cast by a stick tensor should be more significant in the areas that belong to the most likely continuation of the feature that is encoded.

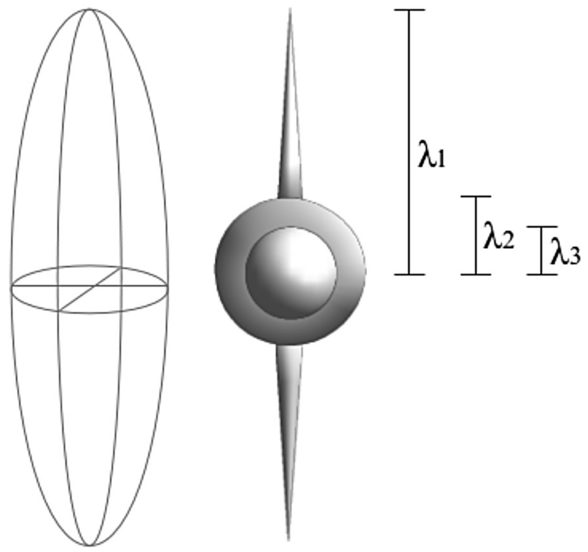


Fig. 6 *In 3D, a symmetric tensor can be decomposed into stick, plate and ball components, with $\lambda_1 - \lambda_2$ its stickness, $\lambda_2 - \lambda_3$ its plateness and λ_3 its pointness*

In all formulations of tensor voting, the direction of the tensor that is cast as a vote at a certain site has a normal lying in the radius of the osculating circle that connects the voter with the vote location. This comes out of the observation that the osculating circle [10] represents a smooth continuation of an oriented feature. This is illustrated in Fig. 7. The received vote is rotated an angle of 2θ with respect to the voter. The original formulation crops the votes at locations outside $-\pi/4 \leq \theta \leq \pi/4$.

The 'saliency' of the vote is penalised with distance along the osculating circle to reduce the correlation between positions far apart. It is also penalised with curvature, so that straight line continuations are preferred. The resulting decay function is shown in the following equation

$$\mathrm{DF}(\boldsymbol{v}) = e^{-(s^2 + c\kappa^2/\sigma^2)} \tag{5}$$

where s is the arc length along the osculating circle and κ is the curvature, which can be computed after $\boldsymbol{v}$. σ is the scale parameter, related to the neighbourhood size: a larger σ should be used to infer curves with larger gaps. It has been observed, however, that the framework is not very sensitive to the scale, making it a fairly robust choice. Discussions on the robustness to scale changes have been presented in [6, 11]. c controls the decay of the field with curvature, and it can be optimally adjusted as a function of the scale parameter σ. The expression

$$c = \frac{-16\ln(0.1)x(\sigma - 1)}{\pi^2} \tag{6}$$

is commonly used, which is derived from the intention of assigning equal probabilities to right angle and round corner continuations of two orthogonal line segments [12]. This way, σ constitutes the only free parameter in the framework.

The vote SV cast by a stick tensor $\boldsymbol{T}$ at position $\boldsymbol{v}$ is then expressed as follows

$$\mathrm{SV}(\boldsymbol{T}, \boldsymbol{v}) = \begin{cases} \mathrm{DF}(\boldsymbol{v}) R_{2\theta} \boldsymbol{T} \boldsymbol{R}_{2\theta}^T, & \text{if } -\pi/4 \leq \theta \leq \pi/4 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

with $\boldsymbol{R}_\alpha$ is a rotation matrix for an angle α. The voting tensor is rotated and scaled following the decay function to produce the vote at a particular position in space.

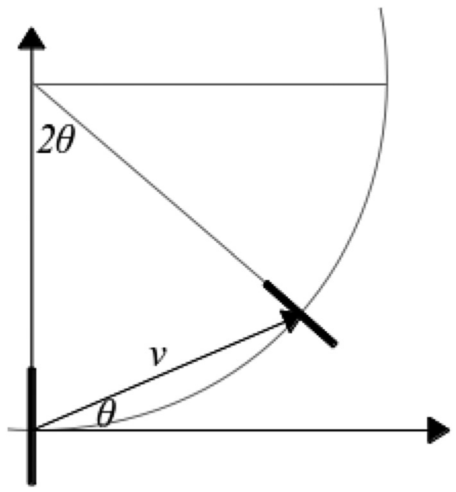


Fig. 7 *Direction of a stick vote points to the centre of the osculating circle*

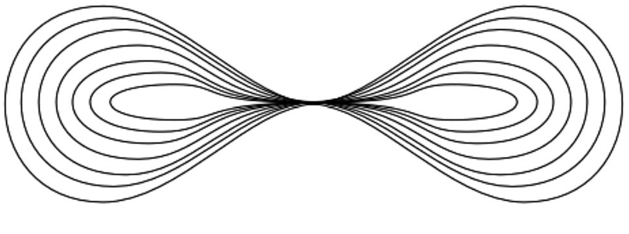


Fig. 8 *Level sets of the saliency of the stick voting field*

In Fig. 8, level sets of the 'saliency' of the stick voting field are illustrated to show the effect of the decay function.

It is important to observe that the voting field in the previous equations is aligned with the voter, so that the vote is expressed in a local coordinate system. The mechanism of vote propagation of the rotated stick fields is depicted in Fig. 9.

In 3D, the computation of the stick voting field is trivial, being a rotation of the flat 2D field around the stick.

2.4 Computing ball and plate votes

The ball voting field can be computed by integrating the contribution of a rotating stick. Since ball tensors encode the uncertainty of orientation in all directions, the stick is rotated around a sphere in 3D. This is illustrated in Fig. 10 where a 'slice' of the 3D field (which spans in fact all around the stick that casts the vote) is seen to rotate with two degrees of freedom.

Let $S(\phi, \omega)$ be a unitary stick tensor oriented in the direction $(1, \phi, \omega)$ in polar coordinates. Note that, we let this tensor have two degrees of freedom in its orientation, which we represent as ϕ and ω. This is observed in Fig. 10, where the field is shown to be rotated in two axes. The vote cast by a ball B can then be expressed in the following way

$$\mathrm{BV}(B, \boldsymbol{v}) = \frac{3\lambda}{4\pi}\int_{\Gamma} \mathrm{SV}(S(\phi, \omega), \boldsymbol{v})\mathrm{d}\Gamma \tag{8}$$

where Γ is the surface of a unitary sphere, λ any of the eigenvalues of B (which are equal) and SV as has been defined in (7).

The plate field computation is analogue. Given that plate tensors encode uncertainty in the orientation around one axis, a stick is rotated around a circumference as shown in Fig. 11, where a slice of the 3D stick field is seen to rotate with one degree of freedom.

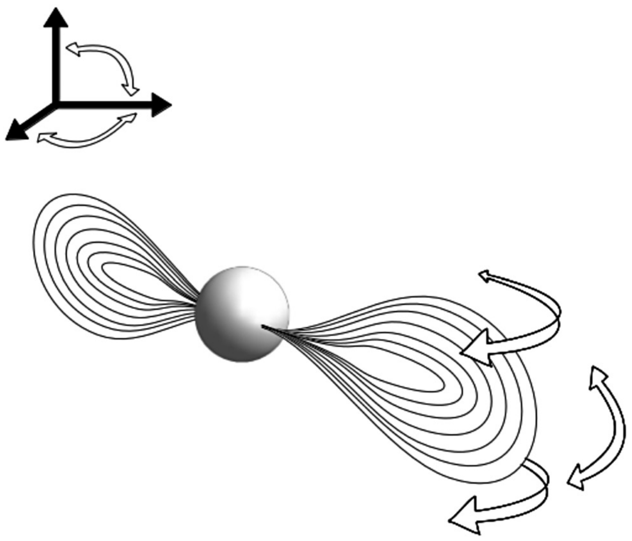


Fig. 10 *Ball field: the fundamental field is rotated and integrated around two axes*

If $S(\beta)$ is now redefined as a unitary tensor rotated an angle β (having one degree of freedom as shown in Fig. 11) a vote cast by a plate tensor P can be constructively defined as follows

$$\mathrm{PV}(P, \boldsymbol{v}) = \frac{\lambda}{\pi}\int_{0}^{2\pi} \mathrm{SV}(S(\beta), \boldsymbol{v})\mathrm{d}\beta \tag{9}$$

λ being one of its equal non-zero eigenvalues.

In the case of the ball voting field, the ballness component is also propagated. From a perceptual point of view, this might not be easily justifiable, as pointed out in [13]. This comes out of the observation that an input ball is used to encode a junction, and junctions are not close to each other. However, this can be useful when tensor voting is used iteratively [14–16] in order to induce uncertainty when tokens are not initialised with accurate values.

2.5 Vote collection and dense feature extraction

Votes are collected by tensorial addition, which is equivalent to adding their matrix representation. In practice, votes with a high percentage of 'saliency' decrease because of distance – usually 99% – are not included. This maximum distance can be dynamically computed after the range parameter σ of (5). This is a common practice to reduce the execution time.

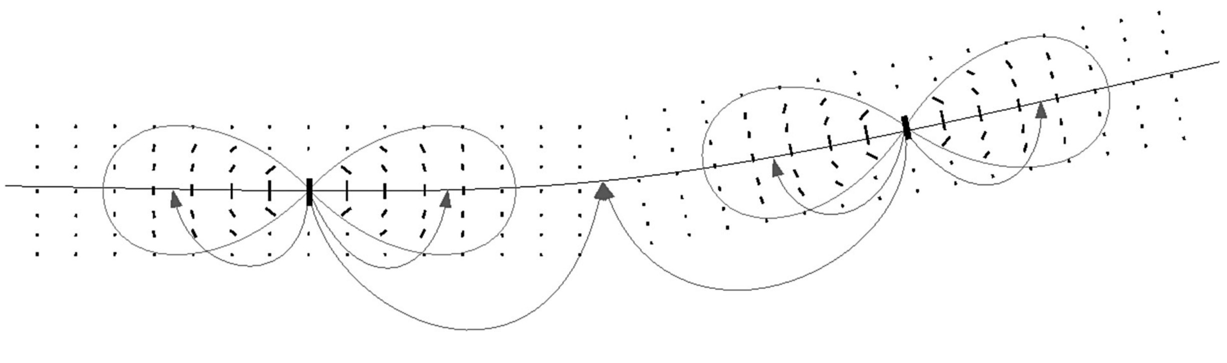


Fig. 9 *Stick voting fields and vote propagation*

Every input token casts votes (the arrows) throughout the space, which are collected and added to infer underlying structure

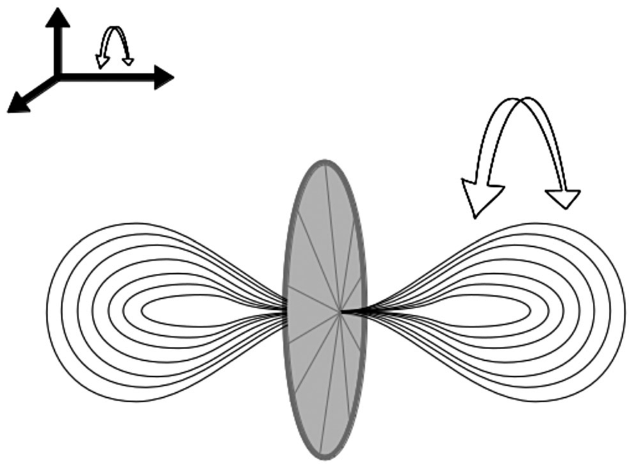


Fig. 11 *Plate field: the fundamental field is rotated and integrated around one axis*

To extract the inferred structures after the dense voting, extremal features in the resulting field are sought.

Junctions or intersections are simply found by extracting the local maxima of the lowest eigenvalue field (the so-called 'pointness' in 3D and 'junctionness' in 2D).

Surfaces are extracted by finding the locations at which the surfaceness is locally maximal along the normal direction encoded by the tensor; that is, any deviation from the surface leads to a lower surfaceness. In 2D, analogously, curves are located at sites where a deviation in the normal direction leads to a decrease in the 'saliency' of the field. This is equivalent to finding the zero-crossings of q, the projection of the gradient of the surfaceness/saliency s along the normal directions

$$q = \nabla s \cdot \hat{\boldsymbol{n}} \tag{10}$$

Curves in 3D are obtained in a similar manner, although they are characterised by curveness extremality along the directions of the plate. In this case, the gradient $\boldsymbol{g}$ of the curveness $(\lambda_2 - \lambda_1)$ should be projected as a vector into the plane generated by $\hat{\boldsymbol{e}}_2$ and $\hat{\boldsymbol{e}}_3$, which is normal to the underlying curve. This vector can be obtained as follows

$$\boldsymbol{q} = R(\hat{\boldsymbol{t}} \times \boldsymbol{g}) \tag{11}$$

where R is a rotation to align with the $\hat{\boldsymbol{e}}_2 - \hat{\boldsymbol{e}}_3$ plane and $\hat{\boldsymbol{t}}$ is the curve's tangent. The extremal feature is located at sites where $\boldsymbol{q} = \boldsymbol{0}$. This means that a point is extremal in curveness when any displacement from the tangential continuation produces a lowering in the curveness map.

The use of Marching Cubes [17, 18], an algorithm to find zero-crossing surfaces in scalar fields, has been suggested to extract the inferred features. They are able to compute the voting field only as needed, because they advance through the zero-crossing surface, analysing the field locally. This way, the execution time is highly reduced. Adaptations of the technique to be used in tensor voting have been described [4].

2.6 N-dimensional (N-D) tensor voting

Tensor voting in more than three dimensions has been applied throughout the literature to solve computer vision problems. For example, optical flow has been formulated as a four-dimensional (4D) tensor voting problem, with two dimensions for the velocity field [19], and texture synthesis has been carried out encoding texture descriptors as multi-dimensional tensors [20].

N-D tensor voting still conveys the principles of smooth continuation of features, although in an *N*-D space. The representation of features can be easily generalised to higher dimensions observing (1) and (3). In these expressions, a linear combination of eigenvalue/eigenvector pairs is present. In N dimensions, such a tensor would be represented as an $N \times N$ symmetric non-negative definite matrix (a hyperellipsoid), generated as in the following expression [21]

$$T = \sum_{d=1}^{N} \lambda_d e_d e_d^T \tag{12}$$

As in (1) and (3), this expression generates every elementary tensor by means of the inner product of the eigenvectors, weighted by the eigenvalue.

The expression can be again rewritten to reveal the underlying features, as in (2) and (4)

$$\begin{aligned} T &= (\lambda_1 - \lambda_2)\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T \\ &\quad + (\lambda_2 - \lambda_3)(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T) \\ &\quad + \cdots + \lambda_N(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T + \cdots + \hat{\boldsymbol{e}}_N\hat{\boldsymbol{e}}_N^T) \\ &= \sum_{d=1}^{N-1}\left[(\lambda_d - \lambda_{d+1})\sum_{k=1}^{d}\hat{\boldsymbol{e}}_d\hat{\boldsymbol{e}}_d^T\right] + \lambda_N\sum_{k=1}^{N}\hat{\boldsymbol{e}}_d\hat{\boldsymbol{e}}_d^T \end{aligned} \tag{13}$$

In the previous equation, it can be observed that there is a parameter λ_N affecting every component (from $\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T$ to $\hat{\boldsymbol{e}}_N\hat{\boldsymbol{e}}_N^T$). Intuitively, this parameter represents the uncertainty in all directions, the so-called ballness or pointness.

The other terms in the previous equation are weighted by a subtraction of successive eigenvalues, and affect the summation of a number of eigenvectors, which are the normals to the feature they encode. For example, in the 3D case, the first component represents a surface, a feature that has a single normal in the direction of $\hat{\boldsymbol{e}}_1$. This is the stick component of the tensor. The following component represents a curve, whose normal is now a plane generated by $\hat{\boldsymbol{e}}_1$ and $\hat{\boldsymbol{e}}_2$. This is the so-called plate component. In an *N*-D space, a tensor is a combination of elementary features, where each feature is described by a number of normals ranging from 1 to N.

In other words, every elementary feature in the decomposed form (13) has a normal hypersurface generated by $\hat{\boldsymbol{e}}_1, \ldots, \hat{\boldsymbol{e}}_j$, with $j \leq N$.

The generalised voting fields are computed in a similar fashion. The stick voting field is equivalent, as expressed in (7), and the others are obtained after it. A generalisation for the derived fields can be easily obtained by observing the way plate and ball votes are generated in 3D, as illustrated in Figs. 10 and 11 and expressed in (8) and (9). In the case of the plate, the fundamental field is rotated with one degree of freedom and, in the case of the ball, it is rotated with two degrees of freedom. In general, for an elementary feature described by j normals, the stick is rotated around the hypersphere centred at the origin and passing through $\hat{\boldsymbol{e}}_1, \ldots, \hat{\boldsymbol{e}}_{j-1}$. The contributions are integrated.

3 Difficulties

The original framework of tensor voting featured some concerns that led researchers to investigate ways to improve the technique. Some of the limitations were merely practical, especially regarding execution time or implementation efforts. A number of reformulations have arisen because of these concerns. Other limitations are theoretical, in the sense that tensor voting lacks some capabilities that might be required for certain applications, which motivated augmentations to the framework.

The aim of this section is to review the difficulties reported throughout the literature concerning the original formulation of tensor voting. A number of reformulations addressing some of these issues will be explained later.

3.1 Scale invariance

Scale invariance [22] is a property referring to the invariance of a function regardless of the metric unit used in the space. This can be interpreted as a graph that scales without changing its shape. The lack of scale invariance property of the original formulation of tensor voting has been pointed out in [13] and implicitly addressed in [23].

Equation (5) involves an arc length parameter s and a curvature parameter κ. Considering the geometric properties of the stick votes, these can be obtained in the following way

$$s = \frac{\theta v}{\sin\theta}, \quad \kappa = \frac{2\sin\theta}{v} \tag{14}$$

where v is the distance between the voter and the receiver. It is observed that s directly depends on the length v so that $s(nv) = n{\cdot}s(v)$. The curvature κ depends inversely on the length v, so that $\kappa(nv) = \frac{1}{n}\kappa(v)$.

Scale invariance would be true for stick votes if $\mathrm{DF}(n\boldsymbol{v}) = \mathrm{DF}(\boldsymbol{v})$ [from (5)] [22]. σ is scaled accordingly in this test because it is supposed that a change in metric units should impact in a proportional change in the scale factor [13]. Analysing the validity of this property

$$\begin{aligned}\mathrm{DF}(n\boldsymbol{v}) &= \mathrm{e}^{-((n^2v^2+(ck^2)/(n^2))/(n^2\sigma^2))}\\ &= \mathrm{e}^{-((v^2+(ck^2)/(n^4))/(\sigma^2))}\end{aligned} \tag{15}$$

it is observed that scale invariance is only true if c is directly related to the fourth power of the distance, which is in general not a common practice in tensor voting and is too far related to the actual nature of the parameters in this formulation.

Scale invariance is a desired property considering that a decay function should be dimensionless [24] because it is a factor that alters the strength and not the unit of the votes. In the original formulation, if c is taken as a constant and a unit in space equals a metre [m], the decay function is measured in a [e^{-1/m^4}] unit. If c directly depends on σ, as in (6), the formulation is still not dimensionless with a unit equal to [e^{-1/m^3}].

Scale invariance would be useful in further mathematical developments in the technique and to facilitate the reusability of precomputed fields that can be scaled [13]. Precomputed voting fields and interpolation are a common practice. However, in the original formulation different fields have to be stored not only for every value of c considered but also for different values of the scale σ, given the lack of scale invariance in the original formulation.

3.2 Closed-form solution

In general, a combination of a finite number of elementary operations constitutes a closed-form solution to a problem, although there is no consensus on the exact definition of this property [25].

Tensor voting in its original formulation is clearly not expressed in a closed-form, as has been pointed out in [21, 26]. This is because of the need for discrete sampling and integration steps in the computation of votes (other than the ones produced by a stick tensor), as explained in Section 2.4.

A closed-form expression is often desired to achieve an efficient solution with less implementation efforts. In addition, it allows the application of a number of mathematical operations, including differential calculus, which otherwise would not be possible [26].

In addition, the use of precomputed fields to reduce execution time becomes impractical in higher dimensions, as has been pointed out in [21]. In D dimensions with k samples per axis, k^D tensors of size $D \times D$ are to be stored, after a precomputation involving a discrete integration on D variables. In addition, as dimensionality grows, not only the storage required becomes impractical but also the likelihood that a precomputed vote will be used decreases. A closed-form solution has been suggested in a recent publication though it has later been shown to be incorrect [27].

3.3 Complexity of dense voting

In some applications, it is desirable to compute the resulting voting field densely, that is, for every point in a grid. This way, tensor voting is applied as a filter whose output is the resulting tensorial map. As explained in Section 2.5, the alternative is to start from a highly salient seed and proceed as a march, computing the field as needed. However, the initial seed might not be available.

In addition, in the most general case the input might be dense, that is, there is a token in every grid position [23]. The complexity for a k-dimensional image of side n becomes $O(n^k)$ and, even in the case of stick voting (which does not require numerical integration) this can be prohibitive considering that rotations and inverse trigonometric functions are required in every vote [13]. For example, the authors of [23] report that the dense stick voting for a 512×512 pixel image took 10,000 s to run on a 2.3 GHz machine.

3.4 Curvature information

Tensor voting enforces the smooth continuation of features. However, curvature information is not propagated. For example, if a considerable portion of a circle is missing, the continuation will not favour the completion of the figure as a circle. This example has been pointed out in [28] and the issue has been also addressed in [16]. Such a curvature propagation might be desirable depending on the context.

3.5 Handling of feature discontinuities

The original tensor voting framework was aimed at robustly detecting curves and surfaces out of possibly noisy or corrupted data, although no handling of the endings or boundaries of these features was provided. This way, if using a marching algorithm to extract the meshes, no information other than junction thresholds could be used to decide the ending of the march. The original formulation is able to encode and measure the sharpness of the features, captured in the junction or pointness maps, but does not provide a way to encode the likeliness of a point to belong to the boundary of an object.

This is, of course, not a concern in closed surfaces or curves. However, in some contexts it might be necessary. This has been pointed out in [6, 29]. In [30], the need for endpoint detection in tensor voting is highlighted in the context of fibre characterisation, which shows up as disconnected curve segments in the images.

4 Tensor voting reformulations

In this section, different reformulations of the technique are reviewed. The criteria used here is to include works that involve a relevant modification in theoretical aspects of the original framework. After explaining the different alternatives presented here, a comparison among them is performed.

Throughout this section, the mathematical formulae have been restructured as compared to the original publications, in order to display them in a coherent way and to facilitate their comparison.

Four high-impact reformulations are reviewed, in decreasing order of complexity. A discussion section to summarise the contribution of every method is included independently after each of them.

4.1 Efficient tensor voting (ETV)

ETV is a reformulation presented in [13] that aims at providing equivalent results to those obtained using OTV, but suppresses the need for discrete sampling and integration. ETV provides substantial changes to the way voting fields are calculated, although the effect is highly similar to the original formulation.

The method redefines stick votes to be scale-invariant and later takes advantage of this property to provide new formulations for plate and ball fields.

4.1.1 Stick votes: The authors propose replacing the decay function (5) with the following expression

$$\mathrm{DF}_{\mathrm{ETV}}(\boldsymbol{v}) = e^{-(s^2/\sigma^2)-c\,\bar{\kappa}^2} \tag{16}$$

with $\bar{k}$ as the normalised curvature [equal to $\sin(\theta)$]. This dimensionless curvature indicator has already been used in other contexts [31] and turns stick voting into a scale-invariant function. The authors suggest that there are many other dimensionless measurements of curvature, although this one is closely related to the definition of curvature and less computationally expensive.

The previous equation redefines the stick voting field, although it preserves the spirit of OTV in the sense that stick votes are penalised with distance and curvature. Now c is used to control the preference for flat surfaces over curved ones.

4.1.2 Plate votes: In OTV, it is stated that plate votes depend indirectly on the distance v from the plate, and the angle γ between $\boldsymbol{v}$ and $\hat{\boldsymbol{e}}_3$, the tangent to the underlying curve. The dependence on the distance v is not direct because the arc length s is used. Considering that stick votes are now scale invariant, (9) can be decomposed into two independent functions

$$\mathrm{PV}(P, \boldsymbol{v}) = \lambda f(\boldsymbol{v}, \gamma, \sigma) H(\gamma, c) \tag{17}$$

where f is a scalar decay function and H is a tensorial function that defines the shape of the resulting tensor independently of the spatial distance.

The scalar function f defines a decay with distance, affected by the scale parameter σ, as follows

$$f(v, \gamma, \sigma) = e^{-(t^2 v^2/\sigma^2)} \tag{18}$$

In this expression, t is a function of γ used to correct the use of distance v instead of the arc length s. t cannot be derived analytically, although its influence in the votes is slight. The following approximation for t is suggested after an experimental evaluation

$$t(\gamma) = \begin{cases} (1+\gamma > \sin(\gamma))/2, & \text{if } -\pi/4 \le \gamma \le \pi/4 \\ 1.033, & \text{otherwise} \end{cases} \tag{19}$$

Now, it remains to define H, the function that describes the shape of the resulting vote, independently of the spatial distance. First of all, it is observed that plate votes do not feature a ball component, hence, H can be decomposed as in the following expression

$$H(\gamma, c) = S_H + P_H \tag{20}$$

with S_H as the stick component and P_H as the plate component of the resulting vote. Taking into account (4), the previous equation can be rewritten as

$$H(\gamma, c) = s_s \hat{\boldsymbol{u}}_1 \hat{\boldsymbol{u}}_1^T + s_p(\hat{\boldsymbol{u}}_1 \hat{\boldsymbol{u}}_1^T + \hat{\boldsymbol{u}}_2 \hat{\boldsymbol{u}}_2^T) \tag{21}$$

where $\hat{\boldsymbol{u}}_i$ are the eigenvectors of H and s_s and s_p are the scalar functions that reflect how stick and plate components are affected depending on γ and c.

The eigenvectors $\hat{\boldsymbol{u}}_i$ can be easily computed observing the geometrical relations between the voter and the receiver, by means of rotations an outer product. We refer the reader to [13] for further details on this.

The remaining question is: How do the stick and plate components relate to γ and c? The answer to this question would define the functions s_s and s_p needed to complete the formulation of plate votes. This can be answered by extracting the eigenvalues of H in the following expression, derived from (17)

$$H(\gamma, c) = \frac{\mathrm{PV}(P, \boldsymbol{v})}{\lambda f(\boldsymbol{v}, \gamma, \sigma)} \tag{22}$$

where PV is computed from the OTV formulation (9) in an accurate way, this is, with a small integration step, in order to keep ETV closely related to the original formulation.

s_s and s_p are the functions that capture all the non-linearities of the technique that have been ignored until this point and they cannot be analytically simplified. As was expected, the curves s_s and s_p behave very differently for values of γ inside and outside $[-\pi/4, \pi/4]$. This is because all the sticks contained in the plate cast votes inside the cone that spans an angle $-\pi/4 \le \gamma \le \pi/4$, but outside this cone the number of sticks that cast votes decreases progressively until reaching minima at $\gamma = \pm\pi/2$. Because of this, inside the aforementioned cone the votes are more similar to a plate (and a perfect plate at $\gamma = 0$) and they resemble a stick outside the cone. In addition, the non-linear effect of the arc length s is present throughout all the curves.

This is illustrated in Fig. 12 where the values of s_s and s_p are plotted as a function of γ, and c is set to zero. This can

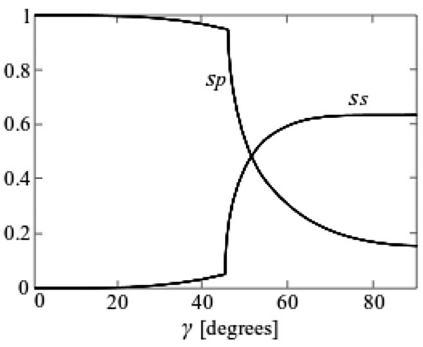


Fig. 12 *Non-linear evolution of plate (s_p) and stick (s_s) components in votes cast by plate tensors*

be interpreted as the level of similarity of a vote to a plate or a ball depending on the angle, when the curvature is not penalised (given that $c = 0$). It can be observed that votes are very similar to a plate (s_p close to 1), with a non-linear decrease of this status until $\gamma = \pi/4$, where the non-linear descent becomes more aggressive. A higher c can be used to accelerate the transformation of the votes into stick-like tensor. In [13], this graph is reproduced for various values of c.

The authors of ETV proposed to perform an experimental fitting of univariate functions on s_s and s_p to obtain expressions that lead to curves that highly resemble the shape of s_s and s_p. Results are available in [32]. The functions obtained are good enough to mimic the behaviour of OTV in votes cast by plate tensors.

4.1.3 Ball votes: The votes cast by a ball depend on a single parameter, the distance v, for a certain combination of σ and σ and c. This is because of the isotropy of the ball voting field. As v increases, the resulting tensors are flattened in one direction, resembling an oblate spheroid. The direction in which this flattening occurs (i.e. the lowest eigenvector of the resulting tensor) is parallel to $\boldsymbol{v}$.

This way, the vote cast by a ball B at $\boldsymbol{v}$ as expressed in (8) can be reformulated in the following way

$$\mathrm{BV}(B, \boldsymbol{v}) = \lambda\left[s_\mathrm{m}\left(\boldsymbol{I} - \frac{\boldsymbol{v}\boldsymbol{v}^T}{v^2}\right) + s_\mathrm{b}\boldsymbol{I}\right] \tag{23}$$

where λ is again any of the equal eigenvalues of B and $\boldsymbol{I}$ is the identity matrix. In the previous expression, $\boldsymbol{I} - (\boldsymbol{v}\boldsymbol{v}^T)/(v^2)$ generates a plate tensor normal to $\boldsymbol{v}$ (an effect achieved by the subtraction of a unit stick tensor from $\boldsymbol{I}$). This way, a perfect plate tensor is generated, a totally flat component. This tensor is multiplied by a factor that we call s_m because it alters the 'magnitude' of the resulting vote. In addition, a second term incorporates a component that is parallel to $\boldsymbol{v}$. This term includes a factor that we call s_b because it controls the 'ballness' of the tensor: if it is zero, the resulting tensor looks like a perfect plate because it lacks a component orthogonal to it, whereas if it is equal to s_m, the result is a perfect ball, given that all the eigenvectors would be equal.

Thanks to scale invariance, s_m and s_b can be decomposed into Gaussian decay functions on $\boldsymbol{v}$ and factors s'_m and s'_b that depend on c that capture all the non-linearities of OTV that have not yet been considered

$$s_\mathrm{m} = s'_\mathrm{m}\, e^{-(v^2/\sigma^2)}, \quad s_\mathrm{b} = s'_\mathrm{b}\, e^{-(v^2/\sigma^2)} \tag{24}$$

As in the case of plate votes, s'_m and s'_b cannot be analytically simplified, although they can be approximated by a fitting on the curve that results after extracting the eigenvalues of the following expression, derived from (23) and (24)

$$H(c) = \frac{BV(B, \boldsymbol{v})}{\lambda e^{-(v^2/\sigma^2)}} \tag{25}$$

In the previous expression, BV is computed through (8), its original formulation, with a small integration step.

Good approximations for s'_m and s'_b are available in [32] and are in fact calculated in a simpler way than those of plate votes given that in this case they depend on a single parameter c.

4.1.4 Discussion: The formulation of ETV transfers the non-linearities of the original formulation to a different place. It is still a numerical approximation, although the need for discrete integration is now avoided. The authors proposed curves that successfully approximate a set of target functions. The result is a novel formulation that closely imitates the behaviour of OTV in an efficient manner. It does not constitute, however, a closed-form solution, nor is it defined for higher dimensions.

4.2 Simplified tensor voting (STV)

The authors of [13] presented a new formulation of tensor voting called simplified tensor voting (STV) which, instead of aimed at providing an accurate numerical approximation of OTV, it intends to redefine the technique based on the perceptual principles that gear the design of the voting fields.

4.2.1 Stick votes: Stick votes are firstly redefined by replacing (5) by the following decay function

$$\mathrm{DF}(\boldsymbol{v}) = e^{-(v^2/\sigma^2) - c\sin^2(\theta)} \tag{26}$$

This decay function is used in (7) as usual, cropping votes outside $-\pi/4 \le \theta \le \pi/4$. It uses normalised curvature $\sin(\theta) = \bar{\kappa}$, hence achieving a scale-invariant formulation, and it replaces the arc length s of OTV by Euclidean distance v. This comes out of the observation that in OTV c controls curvature and σ controls both distance and curvature, because it affects a term that includes a curvature sensitive parameter, the arc length. In (26), now σ can be used to control only the scale and c the curvature. This constitutes a simplification of OTV.

In addition, the time-consuming inverse trigonometric function required to compute the arc length, as expressed in (14), is no longer required.

This is of course a deviation from the original formulation, although the authors suggest that, given that in $-\pi/4 \le \theta \le \pi/4$ the maximum difference between s and v is only of 5.5%, c can be better used to control the effect of curvature and the meaning of the votes is not substantially changed.

4.2.2 Plate votes: Plate votes depend on the angle γ between the tangent of the curve and the point in which the vote is cast, this is, between $\boldsymbol{v}$ and $\hat{\boldsymbol{e}}_3$. The authors of STV firstly observe the behaviour of votes inside the cone $-\pi/4 \le \gamma \le \pi/4$. They state that from a Gestalt point of view, these votes act like stick votes: they are penalised under the same principles although they propagate curvature instead of surfaceness. In other words, the vote cast by a plate is another plate whose magnitude is reduced with arc length and curvature of the osculating circle connecting both plates, the voter and the receiver. This is analogue to the stick voting field decay. As stated before, however, arc length will be replaced by Euclidean distance in the context of STV.

Outside the aforementioned cone, plate votes become more similar to sticks. This is why the authors redefined plate votes including both a plate and a stick component, with a parameter α_P to control how stick votes influence the plate component. The following expression defines the vote cast by a plate tensor P

$$\mathrm{PV}(P, \boldsymbol{v}) = s_{\mathrm{p}} P_{2\gamma} + \alpha_P s_{\mathrm{s}} \lambda(\hat{\boldsymbol{u}}_1 \hat{\boldsymbol{u}}_1^T) \tag{27}$$

where $P_{2\gamma}$ is the original tensor P rotated 2γ, analogously to the orientation of the stick vote as shown in Section 2.3, λ is one of the equal eigenvalues of P and $\hat{\boldsymbol{u}}_1$ is the orientation of the main eigenvector of the resulting stick, which can be obtained by simple trigonometry.

s_{p} is a function that conveys the decay of the plate component, and s_{s} plays a similar role with the stick part. After observing that they have a mirroring evolution, because the stick component becomes more relevant as the angle γ increases, contrary to the case of the plate, these decay functions are defined as follows

$$s_{\mathrm{s}}(P, \boldsymbol{v}) = \begin{cases} e^{-(v^2/\sigma^2) - c \sin^2 \gamma}, & \text{if } -\pi/4 \le \gamma \le \pi/4 \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

$$s_{\mathrm{p}}(P, \boldsymbol{v}) = \begin{cases} e^{-(v^2/\sigma^2) - c \cos^2 \gamma}, & \text{if } \gamma > \pi/4 \text{ or } \gamma < -\pi/4 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

The mirroring effect is achieved by the use of the cosine instead of the sine in the second expression. The definition of these functions has been derived from the novel stick field definition of STV, as in (26).

Parameter $\alpha_P \in [0, 1]$ should be set depending on the type of problem, data density and level of noise. The stick component of plate votes might lead to errors in curved surfaces, hence in those cases the setting of $\alpha_P = 0$ might be appropriate so that the responsibility of the resulting votes lies entirely on the first term of (27). In flat surfaces, the setting of $\alpha_P = 1$ can be beneficial to the estimation of normals. When both configurations exist, flat and curved surfaces, $\alpha_P = 0$ is suggested by the authors.

4.2.3 Ball votes: Votes cast by a ball tensor can be more easily described, since they constitute an isotropic field. The magnitude of the votes decays with distance v. A ball represents total uncertainty about the normal direction at that point. However, the uncertainty is reduced in the direction of $\boldsymbol{v}$ as the distance increases, given that the receiver location might be at the continuation of a feature from the voter. This way, votes become oblate spheroids as v increases its value.

This is expressed in a similar formulation to that of ETV in (23), that conveys the constructive nature of ball votes with two factors: s_{m} to control the way 'magnitude' decreases with respect to distance, and s_{b} to control the ballness of the resulting votes with respect to distance. In ETV, these functions capture the non-linearities of OTV. In STV, however, these factors are redefined to capture in a simpler way the characteristics of the ball field as previously detailed

$$s_{\mathrm{m}}(B, \boldsymbol{v}) = e^{-(v^2/s^2)}, \quad s_{\mathrm{b}}(B, \boldsymbol{v}) = \alpha_B e^{-(v^2/s^2)} \tag{30}$$

Both expressions are simply Gaussian decay functions with respect to distance v, and $\alpha_B \ge 0$ is a parameter that controls the influence of the ball component in the resulting field.

4.2.4 Discussion: STV conveys, in simpler expressions, the intentions that geared the design of the fields in OTV. It is not, however, an equivalent formulation. The efficiency is mainly increased due to the fact that there is no need for the discrete computation of integrals. Additional improvements have been introduced to further simplify the formulation.

Even though STV fields are not equivalent to those of OTV, the formulation proved to be effective in the tests described in [13].

The implementation efforts of the technique are considerably lowered since there are no numerical approximations of any kind in the computation of the votes.

STV introduces two additional parameters, α_P and α_B, that have to be carefully tuned depending on the application. These parameters allow, however, control of the relevance of different components that constitute the votes in the resulting voting fields.

No adaptation of this scheme to an N-dimensional space was proposed by the authors.

4.3 N-D tensor voting (NDTV)

This formulation [21], which we call N-D tensor voting, simplifies the computation of votes with the goal of providing an efficient extension of the OTV framework to an n-dimensional space. This is done taking into account geometric properties of the votes. The formulation here described is different to that of Section 2.6 that simply generalises OTV to N-D.

The authors observed the unlikeliness of using the traditional numerical scheme in higher dimensions, because either execution time is prohibitive, or look-up tables of cached votes become too large.

In the next paragraphs, the way votes are computed in NDTV is explained, starting with the stick voting field and then showing how the other fields are derived after it.

4.3.1 Stick votes: The votes cast by a stick in an N-dimensional space remain equal to OTV, as expressed in (7). The idea can be directly generalised to higher dimensions, in which votes will still be penalised by arc length and curvature of the osculating circle or hypercircle. This is equal to stick voting in N-D as proposed by the authors of OTV.

4.3.2 Votes by stick tensors: The authors of NDTV redefined the remaining votes based on the addition of a number of elementary stick votes that are extracted out of the voter.

The authors firstly carried out an analysis of the voting. It includes features with a number of equal eigenvalues, this is, perfect tensors. These represent the d-dimensional normal space to the feature encoded, which is denoted as N_s. The authors observe that $\boldsymbol{v}$ can be decomposed into $\boldsymbol{v}_{\mathrm{n}}$ and $\boldsymbol{v}_{\mathrm{t}}$, a normal and tangential component to the normal space N_s, as illustrated in Fig. 13. It is then proposed to define an orthogonal basis $\hat{\boldsymbol{b}}_1, \ldots, \hat{\boldsymbol{b}}_d$ that generates N_s. This basis

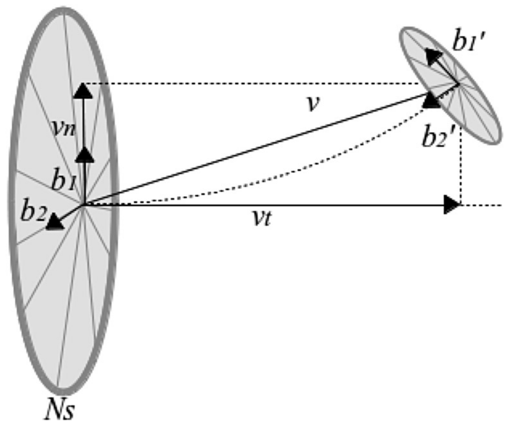


Fig. 13 *Different elements used to construct the vote cast by a plate in NDTV*

must include a component parallel to $\boldsymbol{v}_n$, which is set to be $\hat{\boldsymbol{b}}_1$ for simplicity. The computation of this basis might be done by recurring to a Gram–Schmidt procedure [9].

In the illustrative example of Fig. 13, there is a plate tensor whose normal space is N_s, whose vote at $\boldsymbol{v}$ we wish to compute. Firstly, we must decompose N_s into an orthogonal base. There are infinite possible bases that generate the plate, which are pairs of orthogonal vectors in all possible directions on the plate. However, $\hat{\boldsymbol{b}}_1$ is fixed in the direction of the plane that includes $\boldsymbol{v}_n$ and $\boldsymbol{v}_t$ and the Gram–Schmidt procedure is started with that vector. The orthonormalisation will output $\hat{\boldsymbol{b}}_1$ (or $-\hat{\boldsymbol{b}}_1$).

The received vote can then be constructed taking into account the votes cast by stick tensors coming out of $\hat{\boldsymbol{b}}_1, \ldots, \hat{\boldsymbol{b}}_d$. However, it is remarked that components $\hat{\boldsymbol{b}}'_2, \ldots, \hat{\boldsymbol{b}}'_d$ of the received vote are equal to $\hat{\boldsymbol{b}}_2, \ldots, \hat{\boldsymbol{b}}_d$ except for the fact that they are attenuated only with distance $\boldsymbol{v}_t$. This is because stick votes are independent of curvature when $\theta = 0$. The vote cast by the stick tensor corresponding to $\hat{\boldsymbol{b}}_1$ does depend on curvature and its contribution has to be then fully computed, following the definition of stick votes.

The whole idea can be expressed in the following equation that defines the vote cast by tensor T at $\boldsymbol{v}$

$$V(T, \boldsymbol{v}) = \mathrm{SV}(\hat{\boldsymbol{b}}_1 \hat{\boldsymbol{b}}_1^T, \boldsymbol{v}) + \sum_{i=2}^{d} \mathrm{SV}(\hat{\boldsymbol{b}}_i \hat{\boldsymbol{b}}_i^T, \boldsymbol{v}) \tag{31}$$

where SV is defined in (7) and d is the dimensionality of the normal space N_s of T. The previous expression can be rewritten as follows

$$V(T, \boldsymbol{v}) = \mathrm{SV}(\hat{\boldsymbol{b}}_1 \hat{\boldsymbol{b}}_1^T, \boldsymbol{v}) + \sum_{i=2}^{d} e^{-(v^2/\sigma^2)} \hat{\boldsymbol{b}}_i \hat{\boldsymbol{b}}_i^T \tag{32}$$

considering that all but the first component become plain decay functions on the distance ignoring the curvature.

Resuming the example of Fig. 13, the component $\hat{\boldsymbol{b}}_2$ would cast a vote which is only attenuated with distance, whereas $\hat{\boldsymbol{b}}_1$ will attenuate its contribution both with distance and with curvature.

4.3.3 Ball votes:

Even though ball votes are covered in (32), the authors proposed a simplified equivalent scheme that reduces the number of computations. In fact, (32) also applies for stick votes.

Ball votes include those where $d = n$, that is, all eigenvalues are equal, which constitutes the definition of a ball in an n-dimensional space.

The idea proposed is fairly simple. A ball encodes uncertainty in all directions. The vote cast at a certain location $\boldsymbol{v}$ keeps all the uncertainty of the voter except for the one in the direction of $\boldsymbol{v}$. This is because a straight line connecting both points is considered to be the most likely continuation given that the ball is unoriented, hence normal uncertainty in that direction is entirely suppressed. The way to do this is to build an identity tensor, this is, represented by an identity matrix (which has equal eigenvalues in all directions) and subtract from it a unit stick tensor in the direction of $\boldsymbol{v}$, hence nullifying the uncertainty in that direction

$$\mathrm{BV}(B, \boldsymbol{v}) = e^{-(v^2/\sigma^2)} \left(\boldsymbol{I} - \frac{\boldsymbol{v}\boldsymbol{v}^T}{v^2} \right) \tag{33}$$

where the first factor controls the decay of the field and the second one constructs the appropriate tensor.

4.3.4 Discussion:

The authors proposed a simplified scheme to compute votes whose formulation in N-D is straightforward. In addition, it constitutes an efficient alternative to OTV for two and three dimensions, since it does not require integration.

The formulation is far from OTV, in the sense that it highly simplifies the resulting fields. In OTV, for instance, the plate field does not only include pure plate votes as in NDTV, but an orthogonal component is also present. In the case of balls fields, they contain a ball component in OTV too, which is ignored in this formulation. This does not mean that this is not a sensible choice, for example, as was discussed in Section 2.4, the inclusion of a ball component in ball votes is not easily justifiable from a perceptual point of view, although it might be useful in some contexts. This way, NDTV constitutes a simplified scheme that conveys the majority of perceptual principles of OTV in an efficient manner and with a straightforward extension to N-D, although it is not equivalent to the original formulation.

This formulation is defined for perfect tensors, so (32) cannot be applied to an arbitrary tensor without a prior decomposition (although this might not be necessary at all).

A numerical comparison of this method with OTV is available in [13].

4.4 Steerable tensor voting

This reformulation [23], which we refer to as SteerTV, is not an integral modification to the framework because it only addresses the generation of stick votes. However, the authors managed to reduce the execution time and implementation efforts in a significant way, allowing them to efficiently compute votes densely in a whole image. Considering that in a number of situations only stick votes are the input in tensor voting, this has been a high impact improvement to the framework.

Given that the stick voting fields have to be oriented with the voter, tensor voting cannot be expressed as a traditional convolution operation. However, the authors of this work redefined the stick voting fields in 2D to later express that field as a combination of steerable filters [33, 34].

The redefined decay function that the authors used is as follows

$$\mathrm{DF}(\boldsymbol{v}) = e^{-(v^2/2\sigma^2)} \cos^{2n}(\theta) \tag{34}$$

In the previous expression, n is used to control the penalisation with curvature and σ is the scale factor used throughout the bibliography. It features three key differences compared with OTV:

- Euclidean distance is used instead of arc length.
- A power of cosine is used instead of curvature, turning it invariant to scale (cosine instead of sine is used because the authors express this formulation with tokens that represent tangents instead of normals).
- The votes are not cropped outside $-\pi/4 \le \theta \le \pi/4$, leading to a continuous stick voting field.

After expanding the previous expression, the following complex-valued basis filters are defined (the reader is referred to the original publication for further details on the derivation)

$$w_m(\boldsymbol{x}) = e^{-(x^2+y^2/2\sigma^2)} \left(\frac{x+iy}{\sqrt{x^2+y^2}}\right)^m, \quad \text{for } \boldsymbol{x} \neq (0,\, 0) \tag{35}$$

The value when $\boldsymbol{x} = (0,\, 0)$ is not specified in the original publication. Setting it to zero excludes the centre in the convolution kernel, that is, votes are only computed out of neighbouring contributions. Setting it to 1 includes the centre of the kernel too.

From the input data, a set of complex-valued images are defined at every location $\boldsymbol{x}$ with the following expression

$$c_m(\boldsymbol{x}) = s(\boldsymbol{x}) e^{-im\beta x} \tag{36}$$

where $s(\boldsymbol{x})$ returns the ‘saliency’ at $\boldsymbol{x}$ and $\beta(\boldsymbol{x})$ returns the orientation of the tensor at $\boldsymbol{x}$. Ballness is not considered, given that this formulation is only defined for stick tensors.

Three elements, U_{-2}, U_0 and U_2 are then computed in the following way (n was fixed to 2 for simplicity)

$$\begin{aligned} U_{-2} &= (w_0 * \bar{c}_2) + 4(w_2 * c_0) \\ &\quad + 6(w_4 * c_2) + 4(w_6 * c_4) + (w_8 * c_6) \\ U_0 &= Re(6(w_0 * c_0) + 8(w_2 * c_2) + 2(w_4 * c_4)) \\ U_2 &= \overline{U_{-2}} \end{aligned} \tag{37}$$

In the previous expression, * is a convolution operation, the *bar* operator returns the complex conjugate, $Re(\cdot)$ the real part, and w_{m} and c_{m} have been defined in (35) and (36), respectively.

From the complex-valued fields U_{-2}, U_0 and U_2, the properties of the resulting tensors (‘saliency’, ballness and orientation β) can be derived as follows

$$\begin{aligned} \lambda_1 &= |U_2|, \quad \lambda_1 - \lambda_2 = \frac{1}{2}(U_0 - |U_2|), \\ \beta &= \frac{1}{2}\arg(U_{-2}) \end{aligned} \tag{38}$$

Considering the convolution theorem [35], it can be stated that $a*b = \mathrm{IFFT}(\mathrm{FFT}(a)\cdot\mathrm{FFT}(b))$, that is, the $O(n^2)$ convolution in the space domain can be replaced by a point-wise product in the frequency domain (FFT stands for fast Fourier transform and IFFT for its inverse). This way, the FFTs of the different w_m and c_m can be computed, and U_{-2}, U_0 can be rewritten as pointwise products with one IFFT at the end in the case of U_{-2} (because operations are all linear) and one IFFT in U_0 prior to applying $Re(\cdot)$. This way, the complexity becomes $O(n{*}\log(n))$ because of the computation of the FFT. In the end, the execution time is significantly lowered, as shown in a comparison that is available in [23].

A work generalising this concept to the 3D stick voting field has been published [36]. In this work, the authors follow a similar procedure, making use of tensorial harmonics to expand tensor fields. The mathematical background under these formulations is not as straightforward as the 2D steerable formulation, since expertise in tensorial harmonics is needed, and its implementation does not seem as trivial. However, the authors reported good experimental results considering the 3D setting. We refer the reader to the original publication for a deep explanation on this subject.

4.4.1 Discussion: The authors of SteerTV presented a reformulation that allows the fast computation of dense fields. Even if the stick voting was not a concern if compared to the computation of derived fields that require numerical integration, the stick voting was still a time-consuming task when votes are cast densely.

It is remarkable that the implementation efforts are highly reduced now that no care on orientation or rotations is to be taken and considering that the implementation of (35)–(38) is straightforward. We provide a MATLAB/Octave implementation as supplementary material to this paper with the goal of providing a way to quickly evaluate the behaviour of tensor voting in an oriented 2D input, one of its most common applications [37].

This reformulation is so much faster that it suppresses the need for computing the stick field as needed during the feature extraction stage to improve execution time (as explained in Section 2.5), allowing efficient use of tensor voting as a filter: entire ‘saliency’, ballness and orientation maps are computed for the whole image. In addition, the execution time does not depend on the scale parameter σ, which is a novel contribution of this work.

At this point, we wish to describe an advantage of producing a continuous voting field that has not yet been addressed in the literature. In the original and other derived formulations of tensor voting, votes are cropped outside $-\pi/4 \le \theta \le \pi/4$, as expressed in (7), leading to a discontinuous voting field. When votes are collected, the effect of discontinuities can have a negative impact on the feature extraction process (which was described in Section 2.5). This is because some extra local maxima might exist because of these discontinuities. This is shown in Fig. 14, where OTV has been applied to a set of unoriented input tokens in 2D. In Fig. 14*a*, the resulting saliency field ($\lambda_1 - \lambda_2$) is shown, where higher gray levels represent higher saliency. In Fig. 14*b*, a trivial feature extraction step has been carried out: points are tagged in white if they are locally maximal along the two neighbouring pixels that fall along the normal lines. Many undesired cross-shaped artefacts are revealed in this process, which coincide with the voting field discontinuities of the input tokens. The same elementary feature detection technique was tested with SteerTV and the result, as shown in Fig. 14*c*, is more robust. This reduces the implementation efforts to define

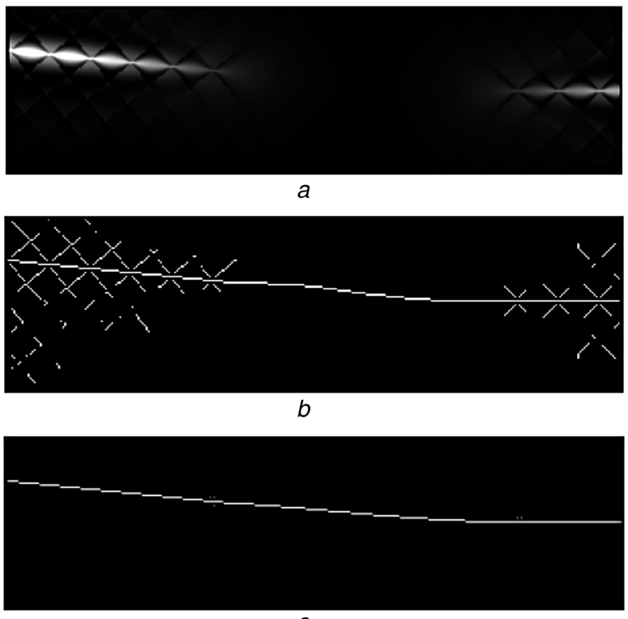


Fig. 14 *Application of OTV to a set of unoriented tokens in 2D*

a Saliency field after voting on a set of unoriented tokens
b Local maxima with OTV
c Local maxima with SteerTV

what constitutes a maximum and the neighbourhood size that is taken into account.

4.5 Comparison

This section includes a comparison of ETV, STV and NDTV. It is intended to provide a quick reference for the reader at the time of selecting a tensor voting approach. A comparative chart is shown in Table 1. The first three columns compare the different alternatives with OTV by '+' and '–' signs, indicating the degree of improvement with respect to the original formulation. Multiple signs (e.g. '++') are used to establish an order in the degree in which every method improves OTV. It does not mean, for instance, that we are assessing that '++' is twice as good as '+'.

In the first column, the equivalence to OTV is depicted. ETV is the most equivalent formulation. STV is considered to be more similar to OTV than NDTV because it allows to set the way the stick component affects a plate vote, or a ball component affects a ball vote. This is not allowed in NDTV, where these components do not even exist in the resulting fields.

It must be pointed out, however, that an exact equivalence to OTV might not be desired. In fact, an experimental study [13] showed better behaviour of STV than OTV. Depending on the application, it might be preferred to use a technique highly similar to OTV or a formulation that somehow conveys the same perceptual principles.

Regarding execution time, all methods represent an advance considering that none of them require numerical integration. STV is a slightly faster choice, from a theoretical point of view, because it avoids the computation of an inverse trigonometric function.

The following column expresses how the reformulation impacts on a lowering of the implementation efforts to use the technique in contrast to OTV. In our comparison, every technique is considered to provide a simpler implementation given that numerical integration or the implementation of vote caches is avoided. ETV is set to be the method that least improves this aspect, given that formulae are quite complex in order to capture the non-linearities of OTV. STV and NDTV both constitute further improvements since the formulations are simplified. No claim of which one constitutes a greater improvement in this aspect is made, given that STV requires more parameter tuning and NDTV requires an algorithm to obtain an orthonormal basis.

The following columns include some other characteristics of every method that might be useful at the time of deciding. First of all, it is stated which method is defined for its use in an *n*-dimensional space, followed by the number of parameters that have to be tuned in every case. STV has two additional parameters (other than σ). However, these allow of control the influence of the different components on the votes.

Finally, the last column details the more time-consuming operation involved in the computation of votes, somehow expanding the data of the second column on efficiency. Considering that millions of votes are sometimes calculated to process a single image, this might be a useful measure of efficiency depending on the application.

Concerning SteerTV, it is not included in the chart because it addresses the computation of dense stick fields exclusively, showing a different intention with respect to the other reformulations and to the original framework itself. It can be, in fact, complimentary to the other methods. However, if we wish to address the same points of the chart, it could be said that SteerTV presents changes with respect to the stick fields of OTV in a similar degree as STV because arc length is replaced by Euclidean distance and curvature is computed with a power of sine. SteerTV improves on the asymptotic complexity of the algorithm regarding stick voting, enhancing execution time even further to the exclusive aim of computing tensor voting for this kind of field. In addition, SteerTV constitutes a distinguishable improvement in implementation efforts since its computation can be expressed as a set of filters and elementary operations. Programming languages or libraries with support for easy matrix operations, like MATLAB or OpenCV, allow SteerTV to be implemented easily.

Concerning the actual execution times, ETV, STV and SteerTV feature comparisons against OTV in their original publications.

Table 1 Comparison of alternative formulations to tensor voting

Method	Equivalence to OTV	Execution time	Implementation efforts reduced	Support for *N*-D	Number of parameters	Highest complexity operations
ETV	–	+	+	no	1	inverse trigonometric function
STV	– –	++	++	no	3	exponential function
NDTV	– – –	+	++	yes	1	inverse trigonometric function

To conclude, we wish to remark that no tensor voting reformulation seems to be 'universal'. A prior application-dependent evaluation has to be done in order to select the most appropriate alternative.

5 Augmentations to the framework

The reformulations presented in the previous section address to some extent the difficulties reported in Section 3, except for the last two, which involve the inclusion of curvature information and the treatment of boundaries and endpoints. These points require an augmentation rather to a reformulation, in order to add new capabilities to the technique. In this section, relevant theoretical advances in augmentations that deal with these two concerns are presented. In addition, we present other relevant works, which extend the capacities of tensor voting to solve particular problems. Depending on the nature of the application, it may or may not be relevant to consider these augmentations.

5.1 Curvature augmentations

In [38], and later in [28], the authors proposed to include an additional voting stage between token refinement and dense voting (Fig. 1). In this stage, curvature information is included in a vector $\boldsymbol{\mu}$ that points to the interior of the curves, and is easily collected through a stick voting process in which a number of $\boldsymbol{\mu}_i$ votes are accumulated.

The direction of the votes $\boldsymbol{\mu}_i$ points to the inner side of the osculating circles throughout the stick voting. This way the accumulation $\boldsymbol{\mu}$ constitutes an estimator of curvature. This is an interesting property *per se*, providing an alternative to other techniques to estimate curvature (see [39, 40] for further details on the subject). In addition, the authors proposed to classify every point into different degrees of curvature and later modify the voting fields accordingly. Depending on the classification, the stick voting might remain equal, curvature can be further penalised (when a locally planar structure is observed) or the field might be entirely cropped to one of its sides (if such a preference is observed in the curvature estimator). This way the later dense voting stage is enhanced, allowing to consider curvature information in the propagation of votes.

A different augmentation to include curvature information was later published in [16]. The authors proposed to directly incorporate the curvature magnitude κ that is computed in the decay function (5) of OTV. This curvature value is accumulated at the receiver in the form of an average, weighted by its saliency. In the next round of voting, the dense stage, the curvature term in the stick fields is affected by this weighted average (it is subtracted from κ). This way, the resulting field is 'tilted' to one of its sides, favouring curved continuation in the direction of the estimated curvature and weighted by its magnitude.

5.2 First-order augmentation

In [29, 41], the authors proposed an extension to the technique that addresses the problem of the treatment of boundaries and endpoints of the detected features. The authors observe that the second-order representation of OTV is not sufficient to encode whether a point is in the interior of a surface or at its boundaries (considering that it is not closed). In addition, it cannot convey whether a point is inside a curve or an endpoint.

It is then proposed to include the so-called 'polarity vectors', which are initialised as zero-valued vectors. The stick voting field is computed after similar principles, although the votes lie in the direction tangential to the osculating circle and point to the voter. The first-order vote is interpreted as an indicator of where the vote is originated in the smooth underlying feature. Decay functions are similar to the second-order vote.

This way, the accumulated votes are cancelled if they come from multiple directions, that is, when the receiver is in the interior of a surface or curve. When the point is at the boundary of a surface or at a curve endpoint, the significance of the vote will be mainly because of the contribution from a subset of directions, pointing to the interior of the feature.

The boundaries and curve endpoints can be extracted as local maxima of the magnitude of the polarity vectors along their directions.

This augmentation is relevant to the practical application of tensor voting, since feature discontinuities are present in many real problems.

In addition, the authors suggest using polarity information to adjust the scale (σ) of the votes. It is proposed to adjust the scale in a fine-to-coarse fashion, until reaching a 'boundariness' lower than a given threshold or reaching a maximum acceptable scale. This is relevant when the density of the data varies throughout the input and a multi-scale voting is convenient for this reason.

5.3 Iterative tensor voting

Even though OTV was originally intended to be a non-iterative technique, in [16], the authors show that the iterative application of tensor voting might be useful for certain cases, including curve reconstruction with large gaps or parameter misadjustment. The successive application of the method can effectively refine the output of each previous iteration, discarding irrelevant components and highlighting the actual underlying structures in highly cluttered scenes.

The main contribution of [16] regarding the iterative scheme is to show that iterations might be useful. The iterative scheme in this work simply repeats the execution of tensor voting a number of times with the same parameters.

Built upon that conclusion, the authors of [14] presented a simple iterative scheme in which tensor voting is applied at multiple scales (i.e. $\sigma \in \sigma_1, \sigma_2, \ldots, \sigma_n$) to effectively extract perceptually prominent elements in cluttered backgrounds and with gaps of different lengths. In the scheme proposed, at each iteration elements below a given threshold at many scales are excluded from the following iteration. Experimental results are available in the publication.

In addition, the same authors presented a different proposal in a later work [42], which iteratively decreases the aperture of the stick voting fields instead of fixing it as a function of the scale [as expressed in (6)]. This has the effect of successively 'funnelling' the field to more precisely infer and connect curves. The authors proved that their method is effective at restoring highly ill-defined curves in the context of synthetic and real medical images.

5.4 Higher-order tensor voting

In Section 2.2, it was stated that the advantage of using second-order tensors has to do with the fact that even under the addition of orthogonal tensors there is a capability for representing 'saliency' information. This extends the limitation of first-order vectors whose magnitude is cancelled upon addition of orthogonal elements.

However, when adding orthogonal second-order tensors, the directional information of the individual components is no longer preserved, increasing the 'ballness' or uncertainty about the orientation. Just like the aim of preserving 'saliency' encouraged the use of second-order tensors, the goal of preserving the directional components after adding tensors leads to think about higher-order tensors. The topic of higher-order tensors for structure estimation has been addressed in [43]. Just like stick second-order tensors can be obtained by computing the outer product of a vector, an order-l stick tensor is obtained by successively computing the outer product l times [43]. In Fig. 15, the notion of a possible sixth-order tensor is depicted, which could be the outcome of the addition of many sixth-order stick tensors.

The authors of [44] presented an extension of tensor voting to infer missing information in higher-order tensor fields, which are widely used to model fibre crossings in a medical imaging modality described in their paper. The authors selected tensor voting to robustly 'inpaint' [20] areas of the images.

The order-l tensor voting proposed in [44] is only defined for stick tensors as an input. The vote propagation principle is very simple: the vote cast is constructed by multiplying a unit tensor by a decay factor. The decay factor is equivalent to that of OTV, as expressed in (5) and the tensor is obtained by computing the outer product of a unit vector in the correct direction l times. This direction follows the same osculating circle approach as expressed in (7). The only difference is the number of times the outer product is computed.

The major point in which higher-order tensor voting differs from OTV is in the interpretation of votes. Now there is not an equivalent decomposition in an orthogonal basis of symmetric tensors, but the algorithm in [45] can be used to decompose the higher-order tensor into a non-orthogonal set of unit vectors $\hat{\boldsymbol{e}}_i$ with corresponding lengths λ_i. Each $\lambda_i \hat{\boldsymbol{e}}_i$ can be interpreted as a stick element in one of the given directions, which now are not cancelled out in the addition of the votes. Now λ_1 and λ_2 can both represent salient elements so that the concept of $\lambda_1 - \lambda_2$ to measure saliency is no longer valid (let us remember that in this approach the tensor decomposition does not lead to orthogonal vectors). The authors proposed to assign high saliency to the directions in which the tensor presents a high convex curvature with respect to the others. We refer the readers to the original paper for further details on how this is accomplished.

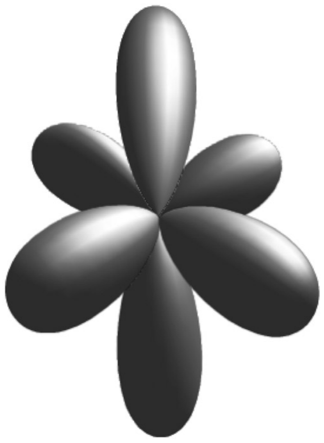


Fig. 15 *Representation of a higher-order symmetrical tensor*

5.5 Probabilistic tensor voting

In [46], the authors presented the so-called probabilistic tensor voting. Firstly, it is observed that the original proposal *per se* is not robust to some types of noise. In particular, OTV does not handle inlier noise in any way, so that no treatment of the possibly incorrect location of a token that does belong to a manifold is performed. This results in the propagation of incorrect information throughout the voting process, even among the members of the actual features.

The solution proposed by the authors is an augmentation of the framework under a simple principle: the location of the voter is extended to be a probability distribution function other than a mere deterministic Cartesian position. The ultimate goal is to compute every vote as the expectation of the different votes cast by a voter located at the different positions according to this probability function.

In practice, the probability distribution functions are considered to a 1D Gaussian curve. In the case of ball votes, this line of possible locations is centred at the input token and its direction is orthogonal to the vector connecting voter and receiver. In the case of stick votes, this distribution spans in the normal direction encoded by the voter. When interpreted as probability distributions, it can be well stated that the mean coincides with the token position and there is a given associated standard deviation.

We here consider this as an augmentation rather than a reformulation because the resulting formulas are simply the same as OTV, although the votes are computed by using a weighted-average of different voters. This leads to modified voting fields, which perform better at inferring structure than traditional OTV.

In addition, the paper builds upon the first-order augmentation, which was presented in Section 5.2 and also describes the formulae which compute first-order components for a probabilistic location of the voter. The authors not only included a polarity vector to detect endpoints, just like in the first-order augmentation, but also included an orthogonal polarity vector, which they call 'error vector', which will eventually point towards the manifold in the presence of inlier noise.

The experimental results of this paper show enhancements against the traditional tensor voting approach in the presence of highly corrupted sparse input tokens.

6 Adaptation of problems to be solved with tensor voting

Considering that tensor voting has been presented as a framework, this section is aimed at showing different ways in which problems have been adapted to be solved by tensor voting. We do not discuss the applications themselves, but we present how the technique has been applied to solve a number of different problems in computer vision.

6.1 Structure-aware tensors

Possibly the most straightforward way of applying tensor voting is by means of structure-aware tensors to infer structural properties of an image. These tensors represent normals to underlying tangible objects in the images. In [7], the authors describe ways of adapting problems to this purpose. A tensorised version of the gradient is frequently used.

Tensor voting as a structure inference engine has been widely used in order to extract salient objects. Some examples are [2, 30, 47, 48]. In the latter, a first-order component was also included to detect endpoints.

Structure-aware tensors have also been used in the context of image 'inpainting' (we refer the reader to [49] for further details on this problem). In [20], the authors infer missing boundaries of objects after an automatic segmentation by means of looking for highly salient points after tensor voting and connecting them with a spline.

In addition to treating natural images, in [50], the authors use structure-aware tensors to extract features directly from triangular meshes, which are the outcome, for example, of a 3D scanning process.

6.2 Tensors in colour space

The technique has also been applied in colour spaces. In this case, some components of the eigensystem of the tensors represent features in a colour space. If this space is designed so that similar colours are close to each other, problems of colour clustering and colour segmentation can be formulated as the finding of saliency surfaces in a tensorial field.

The problem of automatic colour clustering was addressed in [51] by encoding input data into 2D tensors. The authors suggest using coordinates *a* and *b* of a *Lab* colour space [52] to build feature vectors $\vec{\boldsymbol{t}_{i,j}} = [a_{i,j}, b_{i,j}]^T$ for every point *i*, *j*. These vectors are used to construct 2D stick tensors $\boldsymbol{t}_{i,j}\boldsymbol{t}_{i,j}^T$ and are the input tokens for a tensor voting procedure. However, we must point out that this idea does not strictly follow the perceptual laws captured in the design of the fields of OTV. For example, a surface is propagated smoothly in OTV formulation, but the lack of smoothness is not a necessary property in a colour space where edges are natural discontinuities.

In [53], the authors use a simpler feature vector combining the three channels of RGB in the following way: $\vec{\boldsymbol{t}_{i,j}} = [R_{i,j} + G_{i,j}, G_{i,j} + B_{i,j}]^T$. As in the previous case, 2D stick tensors are built and tensor voting is used.

In [54], a 3D surface is defined as $(x, y, H(x, y))$, where $H(x, y)$ is the value of (x, y) in a chromaticity labelled image $(H = 1/3(|R - G| + |G - B| + |R - B|))$. This way, colour clusters appear as layers in the 3D space. 3D tensor voting was then used to carry out colour segmentation.

In [55], the authors encoded colour, edginess and uniformity in a 2D tensor for every colour channel separately in order to robustly detect edges in noisy images. Colour is represented as the angle of the tensor and edginess and uniformity are captured in the eigensystem. This idea was used to the aim of colour segmentation [56] and image denoising [57]. In these works, however, the traditional voting fields are modified to capture perceptual properties particular of colour spaces, which are different to the ones needed to extract surfaces. A detailed explanation of the readaptation of the voting fields is available in [58].

6.3 Stereo matching

The problem of detecting correspondences in pairs of images taken at different angles has been expressed as a 3D surface extraction problem, in which each point is an $[x, y, d]^T$ vector where *d* is a measure of disparity. After observing that actual disparities form a coherent surface in this 3D space, the authors of [59, 60] have encoded these vectors into 3D tensors to extract the correspondencies with a tensor voting procedure.

The same principle has been applied to correlate multiple shots. The authors of [61] suggest generating $[x, y, d]^T$ tokens analysing images pair-wise, to later use 3D tensor voting to extract correspondences. This contrasts with the usual approaches that extract correspondences pair-wise and then combine the results. In this case, more views imply more tokens to the same 3D approach. Further work has been done to improve the reconstruction of surfaces from multiple views with a similar approach in [62].

In addition, the epipolar geometry estimation problem has been reformulated as the finding of the normal of an 8D hyperplane and solved using tensor voting [63].

6.4 Motion analysis

Problems including motion fields have also been adapted to use tensor voting at some of their stages.

In [64], a motion vector field was encoded as 2D tensors and tensor voting was used to remove outliers.

In [65], the authors proposed to encode motion information paired with locations into 4D tensors to do motion analysis by evaluating the resulting fields after a 4D voting process. This way, pairs of shots at different times are compared. The input information is encoded in $(x, y, \boldsymbol{v}_x, \boldsymbol{v}_y)$ tokens, $\boldsymbol{v}$ being the velocity at (x, y). This idea was applied to real images in [66] and also used in [67, 68].

To the aim of better segmenting motion layers, the authors of [19] proposed to later refine the boundaries of the moving regions by a 2D structure tensor voting. The same method of encoding tokens and 4D voting was later used in [69] where the following step of 2D voting was replaced by a graph-cuts technique, enhancing speed.

The previous approaches compare two frames. In [70, 71], time was included as an additional dimension, being the tokens of the form $(x, y, t, \boldsymbol{v}_x, \boldsymbol{v}_y)$, allowing full incorporation of temporal information. This constitutes a spatiotemporal approach to dense motion layer segmentation, which requires 5D tensor voting.

In [53], to the aim of foreground extraction in videos, a robust background colour model is maintained through tensor voting in colour space and a 4D $(x, y, \boldsymbol{v}_x, \boldsymbol{v}_y)$ approach is used to deal with noise and background motion.

Tensor voting has also been applied in the context of video repairing [72]. The centroids (x, y) of moving features are regularised in time *t* by 3D tensor voting on stick tensors created from $[x, y, t]^T$. This way, smooth trajectories in the spatiotemporal domain are obtained.

6.5 Other adaptations

Problems in other contexts have also been formulated in a tensor voting way, which involve requirements of good continuation and proximity.

A method of texture synthesis has been presented [20]. Textural information is encoded into *N*-D tensors following the intensities in a lexicographical order of patches around every point. NDTV is then used to extrapolate textures.

The problem of dimensionality estimation has also been addressed with a tensor voting approach [21]. This is the problem of estimating the dimensionality *d* of a feature in an $\mathbb{R}^n$ space, where $d \leq n$. For example, a set of points can be arranged in a plane in 3D. However, this might be hard to detect in the presence of outliers. An N-D voting is done out of a cloud of points and the different $\lambda_i - \lambda_{i+1}$ of the resulting field are evaluated.

7 Conclusions and future work

Tensor voting is a computational framework to solve computer vision problems that are related to perceptual grouping. It has been used in very recent work with successful results. For example, the authors of [2] recently used tensor voting to infer structure from ultrasonic medical images. However, the authors did not employ the original formulation of tensor voting. After pointing out some issues, they selected a more recent formulation which reinvented the method while solving some of the issues regarding the original proposal. The experience of these authors is similar to what we faced when we first tried to apply tensor voting: a number of issues arise which make the original proposal not entirely applicable. However, the robust results reported in other works are appealing enough to try to incorporate the framework in our own work.

In this paper, we have attempted to provide a comprehensive survey of theoretical research in the area. In particular, we addressed the original formulation to later review a number of reformulations or augmentations to the framework. In addition, we provided the reader with a guideline on how different authors have adapted traditional computer vision problems to be solved with this technique. Furthermore, as notation was not consistent among the different works, we have unified it throughout this paper to allow better comparison of the different proposals.

Throughout our research, we have also identified a number of possible future directions of work on the matter. There is a first family of possible directions, which has to do better comparing the reformulations with the original framework. Firstly, for SteerTV, we suggest adopting the methodology of evaluation of STV and ETV [13] to assess angular errors against the results of applying the original formulation on a given dataset. This experiment has not yet been carried out for this reformulation and it would be important in order to evaluate the behaviour of the modified decay functions.

Some of the works (STV and SteerTV) build their reformulation upon replacing arc length with Euclidean distance. We suggest experimentally evaluating the impact of the usage of this measure, both in execution time and quality. If quality increases considerably when using the original distance measure, but execution time is a concern, we also suggest considering the use of approximations of inverse trigonometric functions (see for instance [73]).

A second family of future directions of work has to do with the development of new tensor voting formulations that mutually combine the virtues of the different works on the matter. For instance, in NDTV the authors used the original stick field approach but simplified the rest of the fields. The effect of incorporating a different formulation for the original stick field (for instance, the one of STV) combined with the NDTV proposal for the derived fields could be explored.

Except for NDTV and the trivial extension of OTV, the reformulations provide solutions for either 2D or 3D spaces. We must point out that the contribution of NDTV is the only efficient formulation of N-dimensional tensor voting which we are aware of. However, there do not seem to be theoretical bounds that prevent the other reformulations to be extended to higher dimensional spaces. We suggest that further work could be done in order to extend the principles of the other reformulations to higher dimensional spaces. The combination of NDTV with other approaches could be considered as soon as the simplifications of the components of NDTV votes are accepted.

Regarding SteerTV, not only the reformulation does not apply for any dimensional space, but is also restricted to stick votes. Depending on the application, this might not be a concern. For example, if the input tokens are oriented, no ball voting will possibly be needed. However, future research can be done to extend the idea to other voting fields, like plate votes, as well as to generalisations in higher dimensions.

Regarding the tensor voting augmentations, the inclusion of a first-order component to effectively detect feature endpoints has considerably enhanced the capabilities of the framework. However, their computation is still done in the traditional manner, that is, the first-order ball voting fields are computed by means of a numerical integration on the fundamental first-order stick field, analogously to the second-order case. We suggest incorporating some of the lessons learned to compute the derived first-order fields more efficiently. One possible straightforward focus for future research is the derivation of a steerable filter formulation for the first-order stick fields.

Many computer vision problems can be posed in terms of perceptual grouping and tensor voting has been a successful choice in many cases. We expect that this document will constitute a reference on the new approaches to the technique for those who want to apply it, as well as a motivation to do further research on the matter.

8 References

1 Wertheimer, M.: ‘Laws of organization in perceptual forms’, Source Book of Gestalt Psychology, London, Routledge & Kegan Paul, 1938, pp. 71–88

2 Martinez-Sanchez, A., Garcia, I., Asano, S., Lucic, V., Fernandez, J.-J.: ‘Robust membrane detection based on tensor voting for electron tomography’, *J. Struct. Biol.*, 2014, **186**, (1), pp. 49–61

3 Guy, G., Medioni, G.: ‘Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 1997, **19**, (11), pp. 1265–1277

4 Tang, C.-K., Medioni, G.: ‘Inference of integrated surface, curve and junction descriptions from sparse 3d data’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 1998, **20**, (11), pp. 1206–1223

5 Medioni, G., Lee, M.S., Tang, C.K.: ‘A computational framework for feature extraction and segmentation’ (Elsevier, 2000), ch. 4

6 Medioni, G., Mordohai, P.: ‘Emerging topics in computer vision’ (Prentice-Hall, 2005), ch. 5

7 Moreno, R., Pizarro, L., Burgeth, B., Weickert, J., Garcia, M.A., Puig, D.: ‘Adaptation of tensor voting to image structure estimation’, in Laidlaw, D.H., Vilanova, A. (Eds.): ‘New Developments in the Visualization and Processing of Tensor Fields, Mathematics and Visualization’, (Springer, Berlin Heidelberg, 2012), pp. 29–50

8 Medioni, G., Tang, C.-K., Lee, M.-S.: ‘Tensor voting: theory and applications’. Proc. of Reconaissance des Formes et Intelligence Artificielle In cole Nationale Suprieure des Tlcommunications, editor, February 2000

9 Bj, A.: ‘Numerics of Gram-Schmidt orthogonalization’, *Linear Algebr. Appl.*, 1994, **197–198**, pp. 297–316

10 Farin, G., Sapidis, N.: ‘Curvature and the fairness of curves and surfaces’, *IEEE Comput. Graph. Appl.*, 1989, **9**, (2), pp. 52–57

11 Medioni, G., Lee, M.S., Tang, C.K.: ‘A computational framework for feature extraction and segmentation’ (Elsevier, 2000) ch. 5

12 Guy, G., Medioni, G.: ‘Inferring global perceptual contours from local features’, *Int. J. Comput. Vis.*, 1996, **20**, (1–2), pp. 113–133

13 Moreno, R., Garcia, M.A., Puig, D., Pizarro, L., Burgeth, B., Weickert, J.: ‘On improving the efficiency of tensor voting’, *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011, **33**, (11), pp. 2215–2228

14 Loss, L., Bebis, G., Nicolescu, M., Skurikhin, A.: ‘An iterative multi-scale tensor voting scheme for perceptual grouping of natural

shapes in cluttered backgrounds', *Comput. Vis. Image Underst.*, 2009, **113**, (1), pp. 126–149

15 Fischer, S., Bayerl, P., Neumann, H., Redondo, R., Cristbal, G.: 'Iterated tensor voting and curvature improvement', *Signal Process.*, 2007, **87**, (11), pp. 2503–2515

16 Fischer, S., Bayerl, P., Neumann, H., Cristbal, G., Redondo, R.: 'Are iterations and curvature useful for tensor voting?', in Pajdla, T., Matas, J. (Eds.): 'Computer Vision–ECCV 2004', (*LNCS* 3023) (Springer, Berlin, Heidelberg, 2004), pp. 158–169

17 Lorensen, W.E., Cline, H.E.: 'Marching cubes: a high resolution 3d surface construction algorithm'. Proc. of the 14th Annual Conf. on Computer Graphics and Interactive Techniques, SIGGRAPH'87, ACM, 1987, pp. 163–169

18 Thirion, J.-P., Gourdon, A.: 'The 3d marching lines algorithm', *Graph. Models Image Process.*, 1996, **58**, (6), pp. 503–509

19 Nicolescu, M., Medioni, G.: 'Motion segmentation with accurate boundaries – a tensor voting approach'. Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 2003, vol. 1, pp. 382–389

20 Jia, J., Tang, C.-K.: 'Inference of segmented color and texture description by tensor voting', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2004, **26**, (6), pp. 771–786

21 Mordohai, P., Medioni, G.: 'Dimensionality estimation, manifold learning and function approximation using tensor voting', *J. Mach. Learn. Res.*, 2010, **11**, pp. 411–450

22 Sornette, D.: 'Critical phenomena in natural sciences' (Springer, 2006), ch. 5.3

23 Franken, E., Almsick, M., Rongen, P., Florack, L., Romeny, B.H.: 'An efficient method for tensor voting using steerable filters', in Leonardis, A., Bischof, H., Pinz, A. (Eds.): 'Computer Vision–ECCV 2006', (*LNCS* 3954) (Springer, Berlin Heidelberg, 2006), pp. 228–240

24 Gibbings, J.C.: 'Dimensional Analysis' (Springer, 2011)

25 Borwein, J.M., Crandall, R.E.: 'Closed forms: What they are and why we care', *Not. Am. Math. Soc.*, 2013, **60**, (1), pp. 50–65

26 Wu, T.-P., Yeung, S.-K., Jia, J., Tang, C.-K., Medioni, G.: 'A closed-form solution to tensor voting: theory and applications', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2012, **34**, (8), pp. 1482–1495

27 Maggiori, E., Manterola, H.L., del Fresno, M.: 'Comments on "A closed-form solution to tensor voting: theory and applications"', *IEEE Trans. Patt. Anal. Mach. Intell.*, DOI:10.1109/TPAMI.2014.2342233

28 Tang, C.-K., Medioni, G.: 'Curvature-augmented tensor voting for shape inference from noisy 3d data', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002, **24**, (6), pp. 858–864

29 Tong, W.-S., Tang, C.-K., Mordohai, P., Medioni, G.: 'First order augmentation to tensor voting for boundary inference and multiscale analysis in 3d', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2004, **26**, (5), pp. 594–611

30 Strokina, N., Kurakina, T., Eerola, T., Lensu, L., Klviinen, H.: 'Detection of curvilinear structures by tensor voting applied to fiber characterization', in Kamarainen, J.-K., Koskela, M. (Eds.): 'Image Analysis', (*LNCS* 7944) (Springer, Berlin Heidelberg, 2013), pp. 22–33

31 Andrews, E.W., Gibson, L.J.: 'The role of cellular structure in creep of two-dimensional cellular solids', *Mater. Sci. Eng. A*, 2001, **303**, (1–2), pp. 120–126

32 Appendix of on improving the efficiency of tensor voting. Available at the Computer Society Digital Library

33 Freeman, W.T., Adelson, E.H.: 'The design and use of steerable filters', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1991, **13**, (9), pp. 891–906

34 Perona, P.: 'Deformable kernels for early vision', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1995, **17**, (5), pp. 488–499

35 Katznelson, Y.: 'An introduction to harmonic analysis' (Cambridge University Press, 2004)

36 Reisert, M., Burkhardt, H.: 'Efficient tensor voting with 3d tensorial harmonics'. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops, CVPRW'08, 2008, pp. 1–7

37 Available at http://www.mathworks.com/matlabcentral/fileexchange/47398

38 Tang, C.-K., Medioni, G.: 'Robust estimation of curvature information from noisy 3d data for shape description'. Proc. of the Seventh IEEE Int. Conf. on Computer Vision, 1999, vol. 1, pp. 426–433

39 Flynn, P.J., Jain, A.K.: 'On reliable curvature estimation'. IEEE Computer Society Conf. on IEEE Computer Vision and Pattern Recognition, Proc. CVPR'89, 1989, pp. 110–116

40 Trucco, E., Fisher, R.B.: 'Experiments in curvature-based segmentation of range data', *IEEE Trans. Pattern Anal. Mach. Intell.*, 1995, **17**, (2), pp. 177–182

41 Tong, W.-S., Tang, C.-K., Medioni, G.: 'First order tensor voting, and application to 3-d scale analysis'. Proc. of the 2001 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR, 2001, vol. 1, pp. 175–182

42 Loss, L.A., Bebis, G., Parvin, B.: 'Iterative tensor voting for perceptual grouping of ill-defined curvilinear structures', *IEEE Trans. Med. Imaging*, 2011, **30**, (8), pp. 1503–1513

43 Schultz, T., Weickert, J., Seidel, H.-P.: 'A higher-order structure tensor' (Springer, Berlin/Heidelberg, 2009)

44 Schultz, T.: 'Towards resolving fiber crossings with higher order tensor inpainting'. New Developments in the Visualization and Processing of Tensor Fields, Springer, 2012, pp. 253–265

45 Schultz, T., Seidel, H-P.: 'Estimating crossing fibers: a tensor decomposition approach', *IEEE Trans. Vis. Comput. Graph.*, 2008, **14**, (6), pp. 1635–1642

46 Gong, D., Medioni, G.: 'Probabilistic tensor voting for robust perceptual grouping'. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW), 2012, 2012, pp. 1–8

47 Moreno, R., Garcia, M.A., Puig, D.: 'Graph-based perceptual segmentation of stereo vision 3d images at multiple abstraction levels', in Escolano, F., Vento, M. (Eds.): 'Graph-Based Representations in Pattern Recognition,' (*LNCS* 4538) (Springer, Berlin Heidelberg, 2007), pp. 148–157

48 Risser, L., Plouraboue, F., Descombes, X.: 'Gap filling of 3-d microvascular networks by tensor voting', *IEEE Trans. Med. Imaging*, 2008, **27**, (5), pp. 674–687

49 Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: 'Image inpainting'. Proc. of the 27th Annual Conf. on Computer Graphics and Interactive Techniques, 2000, pp. 417–424

50 Kim, H.S., Choi, H.K., Lee, K.H.: 'Feature detection of triangular meshes based on tensor voting theory', *Comput.-Aided Des.*, 2009, **41**, (1), pp. 47–58

51 Dinh, T.N., Park, J., Lee, C., Lee, G.: 'Tensor voting based color clustering'. IEEE 2010 20th Int. Conf. on Pattern Recognition (ICPR), 2010, pp. 597–600

52 Tkalcic, M., Tasic, J.F.: 'Colour spaces: perceptual, historical and applicational background'. EUROCON Computer as a Tool, the IEEE Region 8, 2003, vol. 1, pp. 304–308

53 Kulkarni, M., Rajagopalan, A.N.: 'Tensor voting based foreground object extraction'. IEEE Third National Conf. on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2011, pp. 86–89

54 Lim, J., Park, J., Medioni, G.G.: 'Text segmentation in color images using tensor voting', *Image Vis. Comput.*, 2007, **25**, (5), pp. 671–685

55 Moreno, R., Garcia, M.A., Puig, D., Julia, C.: 'Robust color edge detection through tensor voting'. 2009 16th IEEE Int. Conf. on Image Processing (ICIP), 2009, pp. 2153–2156

56 Moreno, R., Garcia, M.A., Puig, D.: 'Robust color image segmentation through tensor voting'. 2010 20th Int. Conf. on Pattern Recognition (ICPR), 2010, pp. 3372–3375

57 Moreno, R., Garcia, M.A., Puig, D., Juli, C.: 'On adapting the tensor voting framework to robust color image denoising', in Jiang, X., Petkov, N. (Eds.): 'Computer Analysis of Images and Patterns,' (*LNCS* 5702) (Springer, Berlin Heidelberg, 2009), pp. 492–500

58 Moreno, R., Garcia, M.A., Puig, D., Juli, C.: 'Edge-preserving color image denoising through tensor voting', *Comput. Vis. Image Underst.*, 2011, **115**, (11), pp. 1536–1551

59 Lee, M.-S., Medioni, G.: 'Inferring segmented surface description from stereo data'. Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, 1998, pp. 346–352

60 Lee, M.-S., Medioni, G., Mordohai, P.: 'Inference of segmented overlapping surfaces from binocular stereo', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002, **24**, (6), pp. 824–837

61 Mordohai, P., Medioni, G.: 'Perceptual grouping for multiple view stereo using tensor voting'. Proc. 16th Int. Conf. on Pattern Recognition, 2002, vol. 3, pp. 639–644

62 Wu, T.-P., Yeung, S.-K., Jia, J., Tang, C.-K.: 'Quasi-dense 3d reconstruction using tensor-based multiview stereo'. 2010 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 1482–1489

63 Tang, C.-K., Medioni, G., Lee, M.-S.: 'Epipolar geometry estimation by tensor voting in 8d'. IEEE the Proc. of the Seventh IEEE Int. Conf. on Computer Vision, 1999, vol. 1, pp. 502–509

64 Dinh, T.N., Lee, G.: 'Efficient motion vector outlier removal for global motion estimation'. 2011 IEEE Int. Conf. on Multimedia and Expo (ICME), 2011, pp. 1–6

65 Nicolescu, M., Medioni, G.: 'Perceptual grouping from motion cues using tensor voting in 4-d', in Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (Eds.): 'Computer Vision–ECCV 2002,' (*LNCS* 2352) (Springer, Berlin Heidelberg, 2002), pp. 423–437

66 Nicolescu, M., Medioni, G.: '4-d voting for matching, densification and segmentation into motion layers'. Proc. 16th Int. Conf. on Pattern Recognition, 2002, vol. 3, pp. 303–308

67 Nicolescu, M., Medioni, G.: 'Layered 4d representation and voting for grouping from motion', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2003, **25**, (4), pp. 492–501

68 Nicolescu, M., Medioni, G.: 'A voting-based computational framework for visual motion analysis and interpretation', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2005, **27**, (5), pp. 739–752

69 Dinh, T., Medioni, G.: 'Two-frames accurate motion segmentation using tensor voting and graph-cuts'. IEEE Workshop on Motion and video Computing, WMVC 2008, 2008, pp. 1–8

70 Min, C., Medioni, G.: 'Motion segmentation by spatiotemporal smoothness using 5d tensor voting'. IEEE Conf. on Computer Vision and Pattern Recognition Workshop, CVPRW'06, 2006, p. 199

71 Min, C., Medioni, G.: 'Inferring segmented dense motion layers using 5d tensor voting', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2008, **30**, (9), pp. 1589–1602

72 Jia, J., Tai-Pang, W., Tai, Y.-W., Tang, C.-K.: 'Video repairing: inference of foreground and background under severe occlusion'. IEEE Proc. of the 2004 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR, 2004, vol. 1, pp. I–364

73 Rajan, S., Wang, S., Inkol, R., Joyal, A.: 'Efficient approximations for the arctangent function', *IEEE Signal Process. Mag.*, 2006, **23**, (3), pp. 108–111