On improving the efficiency of tensor voting

Rodrigo Moreno, Miguel Angel Garcia, Domenec Puig, Luis Pizarro,
Bernhard Burgeth and Joachim Weickert

Rodrigo Moreno, Miguel Angel Garcia, Domenec Puig, Luis Pizarro, Bernhard Burgeth and Joachim Weickert, On improving the efficiency of tensor voting, 2011, IEEE Transaction on Pattern Analysis and Machine Intelligence, (33), 11, 2215-2228.
http://dx.doi.org/10.1109/TPAMI.2011.23
Postprint available at: Linköping University Electronic Press
http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-70982

On Improving the Efficiency of Tensor Voting

Rodrigo Moreno, Miguel Angel Garcia, Domenec Puig, Luis Pizarro, Bernhard Burgeth, and Joachim Weickert

***Abstract*—This paper proposes two alternative formulations to reduce the high computational complexity of tensor voting, a robust perceptual grouping technique used to extract salient information from noisy data. The first scheme consists of numerical approximations of the votes, which have been derived from an in-depth analysis of the *plate* and *ball* voting processes. The second scheme simplifies the formulation while keeping the same perceptual meaning of the original tensor voting: the *stick* tensor voting and the *stick* component of the *plate* tensor voting must reinforce surfaceness, the *plate* components of both the *plate* and *ball* tensor voting must boost curveness, whereas junctionness must be strengthened by the *ball* component of the *ball* tensor voting. Two new parameters have been proposed for the second formulation in order to control the potentially conflictive influence of the *stick* component of the *plate* vote and the *ball* component of the *ball* vote. Results show that the proposed formulations can be used in applications where efficiency is an issue, since they have a complexity of order $\mathrm{O}(1)$. Moreover, the second proposed formulation has been shown to be more appropriate than the original tensor voting for estimating saliencies by appropriately setting the two new parameters.**

***Index Terms*—Perceptual methods, tensor voting, perceptual grouping, non-linear approximation, curveness and junctionness propagation.**

I. Introduction

MEDIONI and colleagues [1], [2], [3] proposed tensor voting as a robust technique for extracting perceptual structures from a cloud of points. This technique has been proven versatile, since it has successfully been adapted to problems well beyond the ones to which it was originally applied with excellent results (*e.g.*, [3], [4] and references therein).

Despite its effectiveness, tensor voting cannot be used in applications where efficiency is an issue. This is mainly due to the high computational cost of its classical implementation, especially regarding the *plate* and *ball* tensor voting.

R. Moreno is with the Center for Medical Image Science and Visualization and the Dept. of Medical and Health Sciences, Linköping University, Campus US, 58185 Linköping, Sweden. E-mail: rodrigo.moreno@liu.se.

M. A. Garcia is with the Dept. of Electronic and Communications Technology, Autonomous University of Madrid, Francisco Tomas y Valiente 11, 28049 Madrid, Spain. E-mail: miguelangel.garcia@uam.es.

D. Puig is with the Intelligent Robotics and Computer Vision Group, Rovira i Virgili University, Av. Països Catalans 26, 43007 Tarragona, Spain. E-mail: domenec.puig@urv.cat.

L. Pizarro is with the Dept. of Computing, Imperial College London, 180 Queen's Gate, SW7 2AZ London, UK. E-mail: l.pizarro@imperial.ac.uk. He is also with the School of Informatics Engineering, University Diego Portales, Av. Ejército 441, Santiago, Chile.

B. Burgeth is with the Faculty of Mathematics and Computer Science, Saarland University, Building E2 4, 66041 Saarbrücken, Germany. E-mail:burgeth@math.uni-sb.de.

J. Weickert is with the Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science, Saarland University, Building E1 1, 66041 Saarbrücken, Germany. E-mail: weickert@mia.uni-saarland.de.

This research has been partially supported by the Spanish Ministry of Science and Technology (project DPI2007-66556-C03-03).

This paper proposes two different ways of implementing tensor voting efficiently. The first one is based on a numerical approximation of the *plate* and *ball* tensor voting, which are mainly responsible for the complexity of the original method. The second one is based on a simplified formulation that fulfills the same perceptual rules followed by tensor voting, although reducing its numerical complexity.

This paper is organized as follows. Section II summarizes the original formulation of tensor voting. Section III presents the proposed numerical approach for implementing tensor voting efficiently. Section IV proposes a simplified version of tensor voting based on the perceptual meaning of the *stick*, *plate*, and *ball* tensor voting processes. Section V shows an experimental comparison between the original tensor voting and the two proposed schemes. Finally, Section VI discusses the obtained results and makes some final remarks.

II. Tensor Voting

The formulation of tensor voting presented in this section is different from, although equivalent to, the original formulation in [3]. It has been chosen since it simplifies the descriptions in the following sections.

In 3D, tensor voting estimates saliency measurements of how likely a point lies on a surface, a curve, a junction, or it is noisy. It is based on the propagation and aggregation of the most likely normal(s) encoded by means of tensors through the so-called *stick*, *plate* and *ball* tensor voting.

Tensor voting comprises three stages. In a first stage, a tensor is initialized at every point of the given cloud of points either with a first estimation of its normal, or with a *ball*-shaped tensor if such a priori information is not available. Afterwards, every tensor is decomposed into its three components, namely: a *stick*, a *plate* and a *ball* component. Every component casts votes to the neighboring points by taking into account the information encoded by the voter in that component. Every vote is a tensor that encodes the most likely direction(s) of the normal at a neighboring point. Finally, the votes are summed up and analyzed in order to estimate degrees of surfaceness, curveness and junctionness at every point. Points with low surfaceness, curveness and junctionness are assumed to be noisy observations.

More formally, the tensor voting at $\mathbf{p}$, $\mathrm{TV}(\mathbf{p})$, is given by:

$$\mathrm{TV}(\mathbf{p}) = \sum_{\mathbf{q}\in neigh(\mathbf{p})} \left(\mathrm{SV}(\mathbf{v},\mathrm{S}_{\mathbf{q}}) + \mathrm{PV}(\mathbf{v},\mathrm{P}_{\mathbf{q}}) + \mathrm{BV}(\mathbf{v},\mathrm{B}_{\mathbf{q}}) \right), \tag{1}$$

where $\mathbf{q}$ represents each of the points in the neighborhood of $\mathbf{p}$, SV, PV and BV are the *stick*, *plate* and *ball* tensor votes cast to $\mathbf{p}$ by every component of $\mathbf{q}$, $\mathbf{v} = \mathbf{p} - \mathbf{q}$, and $\mathrm{S}_{\mathbf{q}}$, $\mathrm{P}_{\mathbf{q}}$ and $\mathrm{B}_{\mathbf{q}}$ are the *stick*, *plate* and *ball* components of the tensor

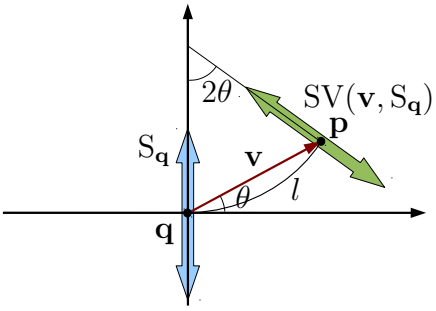

Fig. 1. *Stick* tensor voting. A *stick* $\mathrm{S_q}$ casts a *stick* vote $\mathrm{SV}(\mathbf{v}, \mathrm{S_q})$ to $\mathbf{p}$, which corresponds to the most likely tensorized normal at $\mathbf{p}$.

at $\mathbf{q}$ respectively:

$$\mathrm{S_q} = (\lambda_1 - \lambda_2)\left(\mathbf{e_1}\mathbf{e_1}^T\right), \quad (2)$$

$$\mathrm{P_q} = (\lambda_2 - \lambda_3)\left(\mathbf{e_1}\mathbf{e_1}^T + \mathbf{e_2}\mathbf{e_2}^T\right), \quad (3)$$

$$\mathrm{B_q} = \lambda_3\left(\mathbf{e_1}\mathbf{e_1}^T + \mathbf{e_2}\mathbf{e_2}^T + \mathbf{e_3}\mathbf{e_3}^T\right), \quad (4)$$

where λ_i and $\mathbf{e_i}$ are the i-th largest eigenvalue and its corresponding eigenvector of the tensor at $\mathbf{q}$ respectively.

Saliency measurements can be estimated from an analysis of the eigenvalues of the resulting tensors in (1). Thus, $s_1 = (\lambda_1 - \lambda_2)$, $s_2 = (\lambda_2 - \lambda_3)$, and $s_3 = \lambda_3$ can be used as measurements of surfaceness, curveness and junctionness respectively. Points whose three eigenvalues are small are regarded as noise. In addition, eigenvector $\pm\mathbf{e_1}$ represents the most likely normal for points lying on a surface, whereas $\pm\mathbf{e_3}$ represents the most likely tangent direction of a curve for points belonging to that curve.

The next subsections describe the processes required to calculate *stick*, *plate* and *ball* tensor votes.

A. Stick Tensor Voting

Stick tensors are used by tensor voting to encode the orientation of the surface normal at a specific 3D point. Tensor voting handles *stick* tensors through the so-called *stick* tensor voting, which aims at propagating surfaceness in a neighborhood by using the perceptual principles of proximity, similarity and good continuation borrowed from the Gestalt psychology [5]. The *stick* tensor voting is based on the hypothesis that surfaces are usually smooth. Thus, tensor voting assumes that normals of neighboring points lying on a surface change smoothly. This process is illustrated in Figure 1. Given a known orientation of the normal at a point $\mathbf{q}$, which is encoded by $\mathrm{S_q}$, the orientation of the normal at a neighboring point $\mathbf{p}$ can be inferred by tracking the change of the normal on a joining smooth curve. Although any smooth curve can be used to calculate *stick* votes, a circumference is usually chosen. A decaying function, s_{1s}, is also used to weight the vote as defined below.

For a circumference, it is not difficult to show from Figure 1 that:

$$\mathrm{SV}(\mathbf{v}, \mathrm{S_q}) = s_{1s}\ \left[R_{2\theta}\ \mathrm{S_q}\ R_{2\theta}^T\right], \quad (5)$$

where θ is the angle shown in Figure 1 and $R_{2\theta}$ is a rotation with respect to the axis $\mathbf{v} \times (\mathrm{S_q}\ \mathbf{v})$, which is perpendicular to the plane that contains both $\mathbf{v}$ and $\mathrm{S_q}$. Let $\lambda_{\mathrm{S_q}}$ be the eigenvalue of $\mathrm{S_q}$ greater than zero. The angle θ can be calculated as:

$$\theta = \arcsin\left(\sqrt{\frac{\mathbf{v}^T\ \mathrm{S_q}\ \mathbf{v}}{\lambda_{\mathrm{S_q}}\mathbf{v}^T\mathbf{v}}}\right). \quad (6)$$

A point $\mathbf{q}$ can only cast *stick* votes for $\theta \leq \pi/4$, since the hypothesis that both points $\mathbf{p}$ and $\mathbf{q}$ belong to the same surface becomes more unlikely for higher values of θ. On the other hand, a weighting function, s_{1s}, is used to reduce the strength of the vote with the arc length, l, given by:

$$l = \frac{||\mathbf{v}||\ \theta}{\sin(\theta)}, \quad (7)$$

and with its curvature, κ, given by:

$$\kappa = \frac{2\ \sin(\theta)}{||\mathbf{v}||}. \quad (8)$$

Thus, s_{1s} was defined in [3] as:

$$s_{1s}(\mathbf{v}, \mathrm{S_q}) = \begin{cases} e^{-\frac{l^2+b\kappa^2}{\sigma^2}} & \text{if } \theta \leq \pi/4 \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

where b and σ are parameters. In practice, l ranges from $||\mathbf{v}||$, when $\theta = 0$, to $\frac{\pi}{2\sqrt{2}}\ ||\mathbf{v}|| \approx 1.11\ ||\mathbf{v}||$, when $\theta = \pi/4$.

B. Plate Tensor Voting

Tensor voting utilizes *plate* tensors to encode curves in 3D. Ideally, if a point belongs to a curve, the third eigenvector of its tensor must be aligned with the tangent to the curve at that point, and λ_3 must be zero. Tensor voting handles *plate* tensors through the so-called *plate* tensor voting. Unlike the *stick* tensor voting, whose formulation derives from perceptual rules to propagate surfaceness, *plate* votes are computed in a constructive way. Thus, the *plate* tensor voting uses the fact that any *plate* tensor, P, can be decomposed into all possible *stick* tensors inside the *plate*. Let λ_{i_P} and $\mathbf{e_i}$ respectively be the i-th largest eigenvalue of P and its corresponding eigenvector, R_β be a rotation with respect to an axis parallel to $\mathbf{e_3}$, which is perpendicular to P, and $\mathrm{S_P}(\beta) = R_\beta \mathbf{e_1}\mathbf{e_1}^T R_\beta^T$ be a *stick* inside the *plate* P derived from $\mathbf{e_1}$. Thus, any *plate* tensor P can be written as:

$$\mathrm{P} = \frac{\lambda_{1_\mathrm{P}} + \lambda_{2_\mathrm{P}}}{2\pi}\int_0^{2\pi} \mathrm{S_P}(\beta)\ d\beta. \quad (10)$$

Taking into account that $\lambda_{1_\mathrm{P}} = \lambda_{2_\mathrm{P}}$, and that $\mathrm{S_P}(\beta)$ is a *stick* tensor, the *plate* vote is defined as the aggregation of *stick* votes cast by all the *stick* tensors $\mathrm{S_{P_q}}(\beta)$ that constitute $\mathrm{P_q}$. Thus, the *plate* vote is defined as:

$$\mathrm{PV}(\mathbf{v}, \mathrm{P_q}) = \frac{\lambda_{1_{\mathrm{P_q}}}}{\pi}\int_0^{2\pi} \mathrm{SV}(\mathbf{v}, \mathrm{S_{P_q}}(\beta))\ d\beta. \quad (11)$$

C. Ball Tensor Voting

Ball tensors are utilized by tensor voting to encode junctions or noise. The *ball* tensor voting is defined similarly to the *plate* tensor voting, that is, in a constructive way. Let $\mathrm{S_B}(\phi,\psi)$ be a unitary *stick* tensor oriented in the direction $(1,\phi,\psi)$ in spherical coordinates. Then, any *ball* tensor B can be written as:

$$\mathrm{B} = \frac{\lambda_{1_\mathrm{B}} + \lambda_{2_\mathrm{B}} + \lambda_{3_\mathrm{B}}}{4\pi} \int_\Gamma \mathrm{S_B}(\phi,\psi)\ d\Gamma, \tag{12}$$

where Γ represents the surface of the unitary sphere, and λ_{i_B} are the eigenvalues of B. Taking into account that the three eigenvalues λ_{i_B} are equal, and using the same argument as in the case of the *plate* tensor voting, the *ball* vote is defined as:

$$\mathrm{BV}(\mathbf{v},\mathrm{B_q}) = \frac{3\lambda_{1_{\mathrm{B_q}}}}{4\pi} \int_\Gamma \mathrm{SV}(\mathbf{v},\mathrm{S_{B_q}}(\phi,\psi))\ d\Gamma. \tag{13}$$

III. Efficient Formulation for *Plate* and *Ball* Votes

The evaluation of the *stick* tensor voting is inexpensive, since the rotations involved in that process can be easily avoided by following the geometric constructions of Figure 1. Actually, the complexity of the *stick* tensor voting mainly stems from the computation of an arcsine required to calculate l, and the exponential required by (9). In addition, these computations are not necessary for $\theta > \pi/4$.

Additional efforts have also been made to make the *stick* tensor voting even more efficient. For example, by applying steerable filters in 2D [6], and tensorial harmonics in 3D in order to compute *stick* votes in the frequency domain [7]. Unfortunately, extensions of these methods to calculate *plate* and *ball* votes have not been proposed so far, mainly due to the difficulty to adapt the integrals in (11) and (13) to the frequency domain.

On the other hand, computing *plate* and *ball* votes is highly time consuming, since (11) and (13) cannot be analytically simplified. Thus, researchers usually interpolate precomputed tensor fields in order to reduce the complexity of the *plate* and *ball* tensor voting. Unfortunately, the amount of precomputed information can grow rapidly if several values of parameter b are used, since the voting fields strongly vary with it. In addition, the shape of the voting fields also varies with σ, since (9) is not scale-invariant (cf. Subsection III-A). In practice, this fact involves the use of complex systems for data access and memory management, which are not always available in many applications.

Following a different strategy, [4], [8] and [9] discard part of the votes for the sake of efficiency. Moreover, [10] proposed an efficient implementation of tensor voting that avoids discarding such information through a parallel implementation on a graphics processing unit (GPU). However, the improvement is determined by the number of available processing units. More recently, [11] and [12] proposed a different weighting factor to be used instead of (9) which aims at avoiding its discontinuity. The introduction of this weighting factor simplifies the computations, but at a cost of yielding very different values from those obtained through the original tensor voting.

The following subsections present a numerical approach to implement *plate* and *ball* votes efficiently. Instead of approximating the integrals of equations (11) and (13), the proposed approach is based on the scale-invariant version of *stick* tensor voting described in the following subsection.

A. Scale-Invariant Stick Tensor Voting

Although the formulation of *stick* tensor voting given in Section II is inexpensive, it is not scale-invariant. Scale invariance, which can be thought of as invariance under change of metric units, is a desirable property, since the same results at a particular scale can be obtained for one another by an appropriate scaling of parameters [13]. This property usually followed by physics laws, has been applied to different fields, such as fractal analysis [13], economy [14], and mathematics [15], among many others. Using scale-invariant formulations of tensor voting is advantageous. On the one hand, a scale-invariant tensor voting reduces the complexity of the preprocessing step by only precomputing voting fields at a single scale, since votes at a different scale can be interpolated from the precomputed fields by appropriately scaling spatial distances. On the other hand, a scale-invariant version of the *stick* tensor voting is essential for analyzing the properties of the *plate* and *ball* tensor voting, as shown in the next subsections.

Scale invariance can be defined as follows. Let g be a function of a set of variables, $\mathbf{x}$, which directly depend on the spatial length. Function g is scale-invariant if [13]:

$$g(\mathbf{x}) = g(h\,\mathbf{x}), \text{for any } h \in \Re. \tag{14}$$

This definition can be used to check the scale invariance property of (9). Let us consider s_{1s} in (9) as a function of four variables, namely l, κ, σ and b. Before checking the scale invariance of s_{1s}, it is required to determine the dependency of each variable on the spatial length. First, l and κ directly and inversely depend on the spatial length respectively. Second, σ directly depends on the spatial length, since it is a scale parameter. Finally, b has been chosen in the literature either as a dimensionless constant (e.g., [3], [16]) or as a variable that (mainly) depends on the spatial length (e.g., [17], [4], [8]). It is easy to check that (9) is not scale-invariant under these conditions.

One option to make (9) scale-invariant is by making b dependent on the fourth power of the spatial length, for instance, with b being proportional to σ^4, as proposed in [6]. The main problem with this strategy is the difficulty to set the parameters, since both b and σ determine the influence of curvature in the votes. This paper describes an alternative to assure the scale invariance of (9), keeping intuitive parameter tuning.

In particular, the lack of scale invariance of the *stick* tensor voting is due to the exponent in (9). From dimensional analysis [18], that exponent must be dimensionless in order to assure scale invariance. Thus, (9) can be converted into a scale-invariant equation by using the normalized curvature, $\overline{\kappa}$,

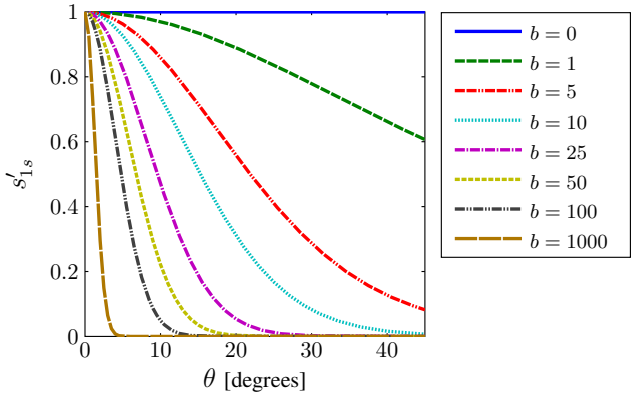

Fig. 2. Evolution of s'_{1s} with respect to θ for some values of b.

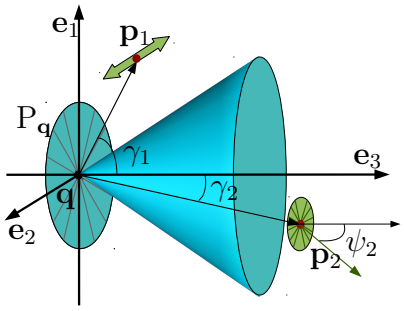

Fig. 3. Examples of the *plate* tensor voting. A tensor $\mathrm{P}_\mathbf{q}$ casts votes to its neighbors with a shape that depends on the cone shown in the figure. Votes are close to *sticks* for points outside the cone ($\mathbf{p}_1$), or close to *plates* for points inside the cone ($\mathbf{p}_2$). Note that neither the *stick* component vanishes inside the cone (except for $\gamma = 0$) nor the *plate* component vanishes outside the cone.

instead of the curvature κ. The normalized curvature is given by [19]:

$$\overline{\kappa} = \kappa \, \frac{||\mathbf{v}||}{2} = \frac{2 \, \sin(\theta)}{||\mathbf{v}||} \, \frac{||\mathbf{v}||}{2} = \sin(\theta), \tag{15}$$

where θ and $\mathbf{v}$ are shown in Figure 1. Since the normalized curvature is dimensionless, it does not require to be weighted by $1/\sigma^2$. Thus, the *stick* tensor voting becomes scale-invariant if (9) is replaced by:

$$s_{1s}(\mathbf{v}, \mathrm{S}_\mathbf{q}) = \begin{cases} e^{-\frac{l^2}{\sigma^2} - b\overline{\kappa}^2} & \text{if } \theta \leq \pi/4 \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

This equation preserves the spirit of (9) in the sense of penalizing *stick* votes by both distance and curvature. Moreover, *plate* and *ball* votes calculated by means of (16) also become scale-invariant thanks to spatial symmetry. Therefore, (16) will be used instead of (9) in the remaining of this work due to its scale invariance.

Figure 2 shows the effect of parameter b on s_{1s}. In this plot, s'_{1s} models the factor of s_{1s} that does not depend on the 3D space, which is given by:

$$s'_{1s} = e^{-b\overline{\kappa}^2}. \tag{17}$$

The figure shows that b can be used to increase the preference for flat surfaces over curved ones. As an example, *stick* votes are negligible when $\theta > 5°$ for $b = 1000$. This means that, in this case, higher values of θ will not be considered to propagate surfaceness. This behavior could be useful to discriminate between flat surfaces and curved ones.

There are many other alternatives of dimensionless measurements of curvature that can be used instead of the normalized curvature. For example, the degree of curvature, which is is given by 2θ, is common in engineering (e.g., [20], [21]) and medical sciences (e.g., [22]). Also, the relative eccentricity, which is given by $(1 - \cos(\theta))/(2\sin(\theta))$, has been used in biomechanics [23]. However, the advantage of using the normalized curvature in tensor voting is that its definition is more closely related to the curvature and it is computationally less expensive than the aforementioned measurements.

B. Efficient Plate Tensor Voting

Scale invariance and spatial symmetry can be used to analyze the *plate* votes. Besides parameters σ and b, the *plate* vote $\mathrm{PV}(\mathbf{v}, \mathrm{P}_\mathbf{q})$ depends on two variables: the distance between $\mathbf{p}$ and $\mathbf{q}$, $||\mathbf{v}||$, and the angle γ between $\mathbf{e_3}$ and $\mathbf{v}$. This angle can be calculated similarly to the angle θ of the *stick* tensor voting:

$$\gamma = \arcsin\left(\sqrt{\frac{\mathbf{v}^T \, \mathrm{P}_\mathbf{q} \, \mathbf{v}}{\lambda_{1_{\mathrm{P}_\mathbf{q}}} \mathbf{v}^T \mathbf{v}}}\right). \tag{18}$$

with $\lambda_{1_{\mathrm{P}_\mathbf{q}}}$ being the largest eigenvalue of $\mathrm{P}_\mathbf{q}$.

The shape of *plate* votes is shown in Figure 3. The first point to remark is that symmetry makes vanish the *ball* component of *plate* votes. This fact has been tested experimentally by checking the coplanarity of the votes cast by every different *stick* inside $\mathrm{P}_\mathbf{q}$. Thus, in general, a *plate* vote can be seen as the summation of a *stick* and a *plate* tensor, each of them with a relative strength that depends on the location of the receiver with respect to the cone of Figure 3.

As shown in that figure, *plate* votes are close to *sticks* for points outside the depicted cone and close to *plates* for points inside the cone ($\gamma \leq \pi/4$). This observation stems from the following reasoning. Recall that a *plate* vote is defined as the summation of the *stick* votes cast by all *sticks* inside the voting *plate*. Let $\mathrm{S}_{\mathrm{P}_\mathbf{q}}(\beta)$ be the *stick* inside the *plate* $\mathrm{P}_\mathbf{q}$ that forms an angle β with respect to $\mathrm{P}_\mathbf{q}\mathbf{v}\mathbf{v}^T\mathrm{P}_\mathbf{q}$, which also lies inside $\mathrm{P}_\mathbf{q}$. The angle θ_β, which is the angle θ required in (5) to rotate $\mathrm{S}_{\mathrm{P}_\mathbf{q}}(\beta)$, can be derived from (6) as:

$$\theta_\beta = \arcsin(|\cos(\beta)| \sin(\gamma)). \tag{19}$$

Thus, θ_β ranges from 0, when $\beta = \pi/2$, to γ when $\beta = 0$. Thus, all *sticks* in the *plate* cast non-null votes for $\gamma \leq \pi/4$, while only those whose $\theta_\beta \leq \pi/4$ cast non-null votes for $\gamma > \pi/4$. An extreme case is given for $\gamma = \pi/2$ where only half of the *sticks* in the *plate* cast non-null votes.

Now, let us consider the case of $b = 0$. In this case, the strength of every *stick* vote is mainly determined by the arc length extended between $\mathbf{p}$ and $\mathbf{q}$ with respect to every voting *stick* $\mathrm{S}_{\mathrm{P}_\mathbf{q}}(\beta)$. For points inside the cone depicted in Figure 3, all $\mathrm{S}_{\mathrm{P}_\mathbf{q}}(\beta)$ cast non-null votes with arc lengths, l, varying from $||\mathbf{v}||$ and $||\mathbf{v}||\gamma/\sin(\gamma)$. Thus, the maximum range of

l is attained for $\gamma = \pi/4$ when it ranges between $||\mathbf{v}||$ and $1.11||\mathbf{v}||$. Thus, for $\gamma \leq \pi/4$, the *stick* component of the *plate* vote is small since the arc length varies in a relatively small range of values. Consequently, the *plate* vote is close to a *plate* inside the cone. For points outside the cone, only a fraction of $\mathrm{S}_{\mathrm{P_q}}(\beta)$ cast non-null votes, since θ_β can be higher than $\pi/4$. This makes some orientations more favored than others, leading to an increase in the *stick* component of the *plate* vote. Thus, the *plate* vote becomes closer to a *stick* outside the cone, although the *plate* component does not completely vanish, even for the extreme case of $\gamma = \pi/2$. This general behavior of the *plate* tensor voting can be modified by using higher values of b. In that case, the transition between the zone where *mainly-plate* and the zone where *mainly-stick* votes are cast is accelerated.

Thanks to scale invariance, the *plate* vote can be divided into two independent functions: a scalar decaying function f, which mainly depends on the spatial distance between the voter and the votee, and a tensorial function H, which does not depend on spatial distance. In practice, f not only depends on $||\mathbf{v}||$ and σ, but also has a slight influence from γ. Thus, (11) can be rewritten as:

$$\mathrm{PV}(\mathbf{v}, \mathrm{P_q}) = \lambda_{1_{\mathrm{P_q}}}\ f(\mathbf{v}, \gamma, \sigma)\ \mathrm{H}(\gamma, b). \tag{20}$$

The scalar function f is given by:

$$f(\mathbf{v}, \gamma, \sigma) = e^{-\frac{t^2 \mathbf{v}^T \mathbf{v}}{\sigma^2}}, \tag{21}$$

where t is a factor that takes into account the use of the arc length l instead of the Euclidean distance in (16). Although t cannot be derived analytically, good approximations can be obtained as follows. As mentioned above, l ranges between $||\mathbf{v}||$ and $||\mathbf{v}||\gamma/\sin(\gamma)$ for $\gamma \leq \pi/4$. Thus, t is bound to the range $[1, \gamma/\sin(\gamma)]$. Thanks to the fact that t varies in a small range of values, the mean arc length, which is given by $||\mathbf{v}||(1 + \gamma/\sin(\gamma))/2$, can be used to approximate t as $(1+\gamma/\sin(\gamma))/2$. Note that t varies in a relatively small range, since $t \in [1, 1.055]$ in this case. On the other hand, if $\gamma > \pi/4$, only a fraction of $\mathrm{S}_{\mathrm{P_q}}(\beta)$ cast non-null votes, which makes it more difficult to find a close approximation for t. The factor t can be experimentally estimated by comparing the trace of *plate* votes computed with arc lengths, as in (9) and (16), to the one computed with Euclidean distances instead. Such experiments yielded that t can be approximated by 1.033 for $\gamma > \pi/4$.

On the other hand, H determines the shape of the *plate* vote. H can be decomposed into its *stick* and *plate* components, whose shapes are shown in Figure 4:

$$\mathrm{H}(\gamma, b) = \mathrm{S_H} + \mathrm{P_H}. \tag{22}$$

These components can be calculated as:

$$\mathrm{S_H} = s'_{1p}\left(\mathbf{u_1}\mathbf{u_1}^T\right), \tag{23}$$

$$\mathrm{P_H} = s'_{2p}\left(\mathbf{u_1}\mathbf{u_1}^T + \mathbf{u_2}\mathbf{u_2}^T\right), \tag{24}$$

with s'_{1p} and s'_{2p} being functions of γ and b, and $\mathbf{u_i}$ being the eigenvectors of H. Functions s'_{1p} and s'_{2p} capture most of the non-linearities involved in (11) and cannot be analytically

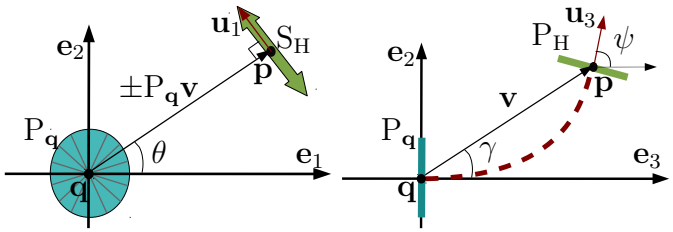

Fig. 4. Components of function H. Left: the *stick* component, $\mathrm{S_H}$, which is always perpendicular to the projection of $\mathbf{v}$ on $\mathrm{P_q}$, given by $\pm\mathrm{P_q}\mathbf{v}$. Right: the *plate* component, $\mathrm{P_H}$, seen from profile. The circumference that joins $\mathbf{q}$ and $\mathbf{p}$ (depicted as a dashed arc) is tangent both to $\mathbf{e_3}$ at $\mathbf{q}$ and to $\mathbf{u_3}$ at $\mathbf{p}$.

simplified. In turn, $\mathbf{u_i}$ can be calculated as follows. Spatial symmetry makes $\mathbf{u_1}$ perpendicular to $\mathbf{e_3}$ and $\mathbf{v}$. Thus,

$$\mathbf{u_1} = \begin{cases} \frac{\mathbf{e_3} \times \mathbf{v}}{||\mathbf{e_3} \times \mathbf{v}||} & \text{if } \mathbf{e_3} \text{ and } \mathbf{v} \text{ are not parallel} \\ \mathbf{e_1} & \text{otherwise} \end{cases} \tag{25}$$

Spatial symmetry also makes $\mathbf{u_3}$ lie on the plane that contains $\mathbf{e_3}$ and $\mathbf{v}$. The angle ψ between $\mathbf{u_3}$ and $\mathbf{e_3}$ is 2γ for $\gamma < \pi/4$, and $\pi - 2\gamma$ otherwise. Thus, $\mathbf{u_3} = R_\psi \mathbf{e_3}$, where R is a rotation with respect to axis $\mathbf{u_1}$. As in the case of the *stick* tensor voting, this rotation can be easily avoided by following the geometry of Figure 4. Having calculated $\mathbf{u_1}$ and $\mathbf{u_3}$, the remaining eigenvector $\mathbf{u_2}$ can be obtained as $\mathbf{u_2} = \mathbf{u_3} \times \mathbf{u_1}$. As stated before, symmetry makes *plate* votes not to have *ball* component. Consequently, H does not to have a *ball* component either, since it models the shape of *plate* votes.

Functions s'_{1p} and s'_{2p} can be estimated from (20) by extracting the eigenvalues of $\mathrm{PV}(\mathbf{v}, \mathrm{P_q})/(\lambda_{1_{\mathrm{P_q}}} f(\mathbf{v}, \gamma, \sigma))$, with $\mathrm{PV}(\mathbf{v}, \mathrm{P_q})$ computed through (11) with a small integration step (e.g., one degree). Figure 5 shows the curves of s'_{1p} and s'_{2p} vs. γ for different values of b. These curves have a discontinuity at $\gamma = \pi/4$. This was expected since the *stick* tensor voting also has a discontinuity at the same angle. The curves corresponding to s'_{1p} and s'_{2p} show that there are mainly two zones in the 3D space depending on which s'_{1p} or s'_{2p} is dominant, which is congruent with the observations made on Figure 3. The curves of s'_{1p} and s'_{2p} also show that b can be used to control the spread of the voting cone of Figure 3, since it becomes narrower as b is increased. These curves can be approximated through any fitting method. As an example, such a fitting can be done by applying two consecutive univariate non-linear fittings as follows:

1) Selection of univariate non-linear functions that mimic the shape of s'_{1p} and s'_{2p} for different fixed values of b. Some preliminary experiments were conducted with different non-linear functions in order to determine suitable non-linear functions to be applied to γ. These functions contain factors c_{1i} and c_{2j}, with $i = 1, \ldots, 6$ and $j = 1, \ldots, 4$, which can be optimized by non-linear least squares fitting.
2) Computation of a non-linear least squares fitting on γ for every different value of b. This process yields a different value of c_{1i} and c_{2j} for every different value of b. At this point, these factors can be stored in order to be queried as look-up tables. These queries are only

required once, since c_{1i} and c_{2j} are only functions of b. However, univariate non-linear fitting of c_{1i} and c_{2j} is an advantageous alternative that avoids the use of look-up tables.

3) Selection of univariate non-linear functions that mimic the evolution of every factor c_{1i} and c_{2j} with b. Some preliminary experiments were conducted in order to determine appropriate non-linear functions to be applied on b.
4) Computation of a non-linear least squares fitting on b for every factor c_{1i} and c_{2j}.

The Appendix shows the fitting yielded by following this methodology. It is important to mention that although more elaborate methodologies can be applied to approximate these curves, the experimental results have shown that their accuracy is good enough to mimic the behavior of the original tensor voting.

The complexity of the proposed implementation of the *plate* tensor voting is mainly due to the computation of an arcsine (which is required to calculate γ), a logarithm required by the approximation of s'_{1p}, and two exponential functions: one for calculating $f(\mathbf{v},\gamma,\sigma)\ s'_{1p}$ and another for calculating $f(\mathbf{v},\gamma,\sigma)\ s'_{2p}$.

C. Efficient Ball Tensor Voting

As in the case of *plate* tensor voting, scale invariance and spatial symmetry can also be used to analyze the *ball* votes. Figure 6 shows some examples of *ball* votes. *Ball* votes are characterized by three properties. The first property is that *ball* votes have an oblate spheroid shape, that is, they are only flattened in one dimension. Thus, $\lambda'_1 = \lambda'_2 > \lambda'_3 > 0$, with λ'_i being the eigenvalues of $\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q})$ in (13). The second property is that the flattened direction of *ball* votes is always parallel to $\mathbf{v}$. This means that the third eigenvector of a *ball* vote, $\mathbf{u_3}$, is parallel to $\mathbf{v}$. The third property is that for some given parameters σ and b, both the size and flatness of *ball* votes only depend on $||\mathbf{v}||$. This condition is given by the isotropic behavior of the *ball* tensor voting. Thus, the *ball* vote can be rewritten as:

$$\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q}) = \lambda_{1_{\mathrm{B}_\mathbf{q}}}\ \left[R_\mathbf{v}\ S(s_{2b},s_{2b},s_{3b})\ \mathrm{B}_\mathbf{q}\ R_\mathbf{v}^T\right], \quad (26)$$

where $R_\mathbf{v}$ is a rotation that makes $\mathbf{u_3}$ and $\mathbf{v}$ parallel, s_{2b} and s_{3b} are functions defined below, and S is a scale transformation that converts the *ball* tensor $\mathrm{B}_\mathbf{q}$ into an oblate spheroid shaped tensor given by:

$$S(s_2,s_2,s_3) = \begin{pmatrix} s_{2b} & 0 & 0 \\ 0 & s_{2b} & 0 \\ 0 & 0 & s_{3b} \end{pmatrix}. \quad (27)$$

The main advantage of (26) is that the expensive integral of (13) is replaced by a rotation. However, this rotation term can also be avoided by constructing the tensor with $\mathbf{v}$. Thus, (26) can be further simplified as:

$$\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q}) = \lambda_{1_{\mathrm{B}_\mathbf{q}}}\ \left[s_{2b}\ \left(\mathrm{I} - \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}\right) + s_{3b}\ \mathrm{I}\right], \quad (28)$$

where I is the identity matrix.

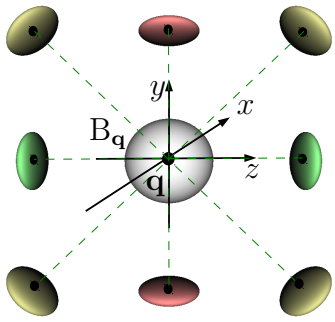

Fig. 6. Examples of the *ball* tensor voting. Point $\mathbf{q}$ casts oblate spheroid shaped votes to its neighbors. $\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q})$ is shown for different positions of $\mathbf{p}$.

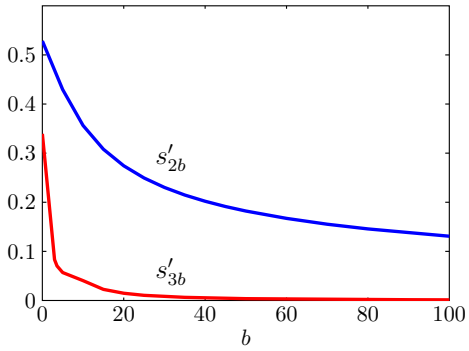

Fig. 7. Evolution of s'_{2b} and s'_{3b} with respect to parameter b.

The purpose of s_{2b} is to control the size of the vote, whereas s_{3b} controls how similar the vote is to a *plate* ($s_{3b} = 0$) or to a *ball* ($s_{3b} = s_{2b}$). Unlike the *plate* tensor voting, isotropy makes functions s_{2b} and s_{3b} only depend on $||\mathbf{v}||$ for specific parameters σ and b. Thus, thanks to the scale invariance, s_{2b} and s_{3b} can be divided into a Gaussian decaying function on $||\mathbf{v}||$ with a standard deviation σ and functions s'_{2b}, s'_{3b} that only depend on b and cannot be analytically simplified, since they capture most of the non-linearities of (13). Thus, s_{ib} is defined as:

$$s_{ib}(\mathbf{v},\sigma,b) = s'_{ib}\ e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2}}, \quad (29)$$

for $i = 2$ and $i = 3$. Factors s'_{2b} and s'_{3b} can be estimated from equations (13), (28) and (29) by following a similar methodology to the one used for factors s'_{1p} and s'_{2p} in the case of the *plate* tensor voting, that is, by extracting the eigenvalues of $\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q})/(\lambda_{1_{\mathrm{B}_\mathbf{q}}} e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2}})$, with $\mathrm{BV}(\mathbf{v},\mathrm{B}_\mathbf{q})$ computed through (13) with a small integration step (e.g., one degree). Figure 7 shows the evolution of s'_{2b} and s'_{3b} with respect to b. It can be seen that $s'_{2b} > s'_{3b}$ for all values of b. This means that *ball* votes are more similar to a *plate* than to a *ball* in general. It can also be seen that parameter b can be used to reduce the size of the *ball* vote. A similar methodology to the one used to approximate s'_{1p} and s'_{2p} can be applied for approximating s'_{2b} and s'_{3b}. In this case, only a single univariate non-linear fitting is required since these functions only depend on b. The Appendix shows the results of least squares fitting for these functions.

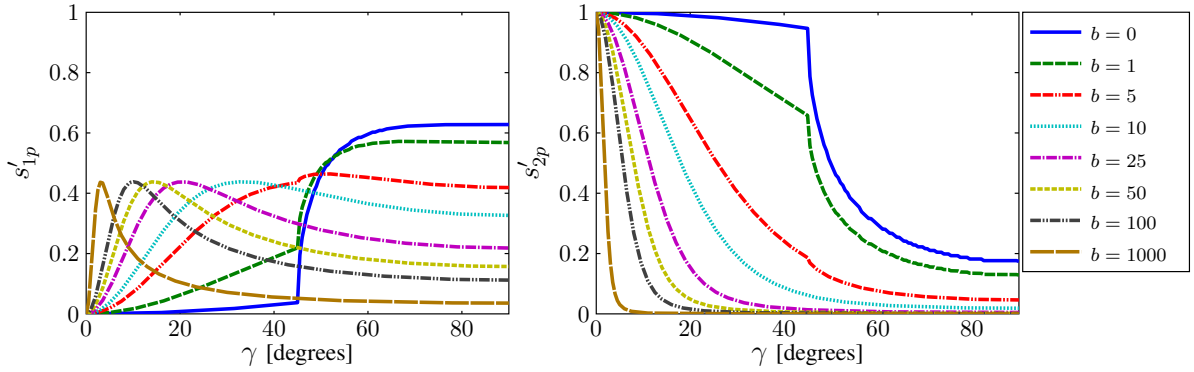

Fig. 5. s'_{1p} and s'_{2p} as functions of γ for different values of parameter b.

The complexity of the proposed implementation of the *ball* tensor voting is mainly due to the computation of a single exponential (required in (29)), since the values of s'_{2b} and s'_{3b} are computed only once.

IV. Simplified Tensor Voting

This section explores an alternative to the numerical approach described in the previous section for calculating tensor voting efficiently. This alternative is based on a simplified formulation that reduces the numerical complexity while keeping the same perceptual rules of tensor voting. The next subsections describe the proposed method to calculate *stick*, *plate* and *ball* tensor votes more efficiently.

A. Stick Tensor Voting

The original *stick* tensor voting can be further simplified while keeping its perceptual meaning by redesigning the weighting function s_{1s} defined in (16). This function has two parameters: b that penalizes the curvature, and σ that penalizes both the distance and curvature (the latter through the $\theta/\sin(\theta)$ factor included in the computation of l in (7)). Thus, for example, it is not possible to avoid the influence of curvature on the calculations, even selecting $b = 0$, since σ not only affects the distance but also the $\theta/\sin(\theta)$ factor, which is related to curvature. For this reason, every parameter has a single task: σ becomes a scale parameter, and b a curvature parameter:

$$s_{1s}(\mathbf{v}, \mathrm{S}_\mathbf{q}) = \begin{cases} e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2} - b\ \sin^2(\theta)} & \text{if } \theta \leq \pi/4 \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

This equation has the additional advantage that the arcsine required for calculating *stick* votes is no longer necessary. Note that the only difference between (30) and (16) is the use of the squared Euclidean distance ($\mathbf{v}^T\mathbf{v}$) instead of the squared arc length (l^2), since $\sin(\theta) = \overline{\kappa}$. This simplification is also based on the fact that the difference between using Euclidean distances and arc lenghts is relatively small. The use of arc lengths can be seen as spatial stretchings of at most 11% (attained at $\theta = \pi/4$) and 5.5% (attained at $\gamma = \pi/4$) for *stick* and *plate* votes respectively. Thus, in general, the effect of the curvature on the votes can be better controlled through b.

B. Plate Tensor Voting

Proposing simplified equations for the *plate* tensor voting requires to understand the perceptual meaning of *plate* votes. From the analysis carried out in Subsection III-B, it can be stated that, from a perceptual point of view, a *plate* vote encodes two different hypotheses, one for every component of the vote.

On the one hand, the hypothesis made by the *stick* component of the *plate* vote is that a neighboring point $\mathbf{p}$ of the voter $\mathbf{q}$ should belong to a surface that abuts the curve that crosses $\mathbf{q}$. However, spatial symmetry makes such a surface be a plane, since the *stick* component is always tangent to the plate $\mathrm{P}_\mathbf{q}$ (see Figure 8). Thus, the *stick* component of the *plate* tensor voting can be thought of as a *stick* tensor voting that makes a stronger hypothesis than the *stick* tensor voting itself, since curved surfaces are only encouraged in the latter. As seen in Figure 8, the *stick* component of the *plate* vote can lead to errors in curved surfaces that must be corrected through *stick* votes cast by other neighbors. This *stick* component mainly appears outside the cone of Figure 3, since points inside the cone can either belong to the curve that crosses $\mathbf{q}$ or to another surface.

In other words, the *plate* tensor voting has less perceptual information to accurately infer a normal at a neighboring point than the *stick* tensor voting. Thus, it is more likely to estimate normals more accurately with the *stick* tensor voting than with the *stick* component of the *plate* tensor voting. However, if no more information is available, for example, if the receiver only gets votes cast by *plates*, the estimation computed by the *plate* tensor voting can be used as the most likely normal at the receiver.

On the other hand, the hypothesis made by the *plate* component of the *plate* vote is that both points, $\mathbf{p}$ and $\mathbf{q}$, should belong to the same curve. In that sense, $\mathbf{p}$ completes the path of the curve that crosses $\mathbf{q}$. This component mainly appears inside the cone of Figure 3, since points outside that cone are more unlikely to belong to the same curve. Thus, the *plate* component of the *plate* vote can be thought of as the natural extension of the *stick* tensor voting in which curveness instead of surfaceness is smoothly propagated by following similar rules. Hence, the *plate* component can be considered to be based on the same Gestalt principles as the *stick* tensor

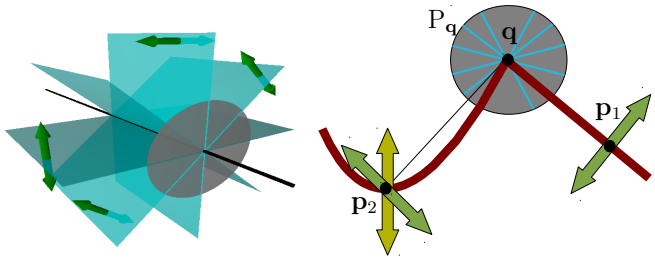

Fig. 8. *Stick* component of the *plate* vote. Left: a curve votes for a plane tangent to the curve. Right: a *plate* tensor in the intersection between a flat and a curved surface (depicted in red). The *stick* component of the *plate* vote can reinforce surfaceness in flat surfaces (see vote at $\mathbf{p}_1$) but also can lead to errors in curved surfaces (see vote at $\mathbf{p}_2$ in green).

voting, namely proximity, similarity and good continuation, but adapted to curveness propagation.

Taking into account these arguments, the following equation is proposed to calculate *plate* votes:

$$\mathrm{PV}(\mathbf{v}, \mathrm{P}_\mathbf{q}) = s_{2p} \ \left[R_{2\gamma} \ \mathrm{P}_\mathbf{q} \ R_{2\gamma}^T\right] + \alpha_P \ \lambda_{1_{\mathrm{P}_\mathbf{q}}} \ s_{1p} \left(\mathbf{u_1}\mathbf{u_1}^T\right), \tag{31}$$

where $\alpha_P \in [0, 1]$ is a new parameter to control the influence of the *stick* component on the *plate* vote, $\lambda_{1_{\mathrm{P}_\mathbf{q}}}$ is the largest eigenvalue of $\mathrm{P}_\mathbf{q}$, $\mathbf{u_1}$ is calculated through (25), $R_{2\gamma}$ is a rotation with respect to $\mathbf{u_1}$, and s_{ip} are weighting functions given by:

$$s_{2p}(\mathbf{v}, \mathrm{P}_\mathbf{q}) = \begin{cases} e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2} - b \ \sin^2(\gamma)} & \text{if } \gamma \leq \pi/4 \\ 0 & \text{otherwise,} \end{cases} \tag{32}$$

$$s_{1p}(\mathbf{v}, \mathrm{P}_\mathbf{q}) = \begin{cases} e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2} - b \ \cos^2(\gamma)} & \text{if } \gamma > \pi/4 \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

Factor s_{2p} in (32) has a similar formulation as s_{1s} in (30), since the *plate* component of *plate* votes is the natural extension to *plates* of the perceptual rules of *stick* tensor voting. In this case, γ is used instead of θ, since curvature is related to the former in *plate* votes. In turn, the *stick* component of *plate* votes has a mirroring evolution with γ when compared to the *plate* component. This inverse relation is modeled in s_{1p} by making it dependent on $\pi/2 - \gamma$ instead of on γ. This is achieved by using $\cos(\gamma)$ instead of $\sin(\gamma)$ in (33). As stated in the previous section, the rotation term can be avoided by following the geometry of Figure 4. Thus, the complexity of this alternative mainly stems from an exponential function for s_{1p} or s_{2p} depending on the angle γ.

The selection of α_P implies a trade-off that depends on the type, density of the data, as well as the level of noise. Thus, by setting $\alpha_P = 0$, the responsibility for estimating normals at surfaces is mainly endorsed to the *stick* tensor voting. This setting should not be used when the data is too sparse to get enough *stick* votes at points in surfaces. In turn, by setting $\alpha_P = 1$, the responsibility for estimating normals at surfaces is shared between the *stick* and the *plate* tensor voting. At flat surfaces, this setting is beneficial since it can help to improve the estimation of normals as more votes are collected, especially in very noisy scenarios. However, at points in curved surfaces, the *stick* component of a *plate* vote can introduce errors whose relevance inversely depends on the number and strength of the *stick* votes cast by other neighbors. For datasets with both flat and curved surfaces, α_P should be set to zero in order to avoid the introduction of errors in the curved surfaces, unless the density of points allowed to cast *stick* votes is large enough to make such an error negligible.

C. Ball Tensor Voting

A perceptual interpretation of *ball* votes, necessary for proposing a simplified *ball* tensor voting, can be obtained from the analysis performed in Subsection III-C. As stated before, a *ball* vote only consists of a *plate* and a *ball* component. On the one hand, the meaning of the *plate* component is that both points, $\mathbf{p}$ and $\mathbf{q}$, should belong to a straight edge in the direction that joins both points. Although a *ball* tensor at $\mathbf{q}$ represents a complete uncertainty about the normal direction at that point, this uncertainty is reduced in the direction $\mathbf{v}$, because both points could belong to a straight edge that is likely joining both points.

On the other hand, the meaning of the *ball* component is that points near a junction should have a junctionness saliency different from zero. From a different point of view, normal uncertainty at a point infers some normal uncertainty at its neighborhood. Unlike the *plate* component, it is difficult to justify from the perceptual point of view the existence of the *ball* component of the *ball* vote since junctions are not usually close to each other. However, it could be useful in iterative schemes *e.g.*, [24], [16], in order to induce uncertainty for those cases in which the tensors are initialized with not too accurate values.

Hence, the same equation (28) is proposed to calculate *ball* votes, but with the following weighting functions:

$$s_{2b}(\mathbf{v}, \mathrm{B}_\mathbf{q}) = e^{-\frac{\mathbf{v}^T\mathbf{v}}{\sigma^2}} \tag{34}$$

$$s_{3b}(\mathbf{v}, \mathrm{B}_\mathbf{q}) = \alpha_B \ s_{2b}(\mathbf{v}, \mathrm{B}_\mathbf{q}), \tag{35}$$

where parameter $\alpha_B \geq 0$ can be used to control the influence of the *ball* component on the *ball* vote. Thus, the high complexity of the *ball* tensor voting is reduced to the computation of a single exponential function. It is important to remark that isotropy makes these functions to be independent from curvature.

A similar reasoning as the one described for the *stick* component of the *plate* tensor voting can be made for the *plate* component of the *ball* tensor voting. The *ball* tensor voting has less information to accurately infer a curve continuation at a neighboring point than the *plate* tensor voting, hence the latter is preferred if available. This would give place to a new parameter to control the influence of the *plate* component of the *ball* vote on the estimation of surface intersections. However, in practice, tensor voting is usually run as proposed in [3], that is, tensors are initialized with unitary *balls* and two rounds of tensor voting are applied: the first one only considers the *ball* tensor voting, whereas the second round only considers both the *stick* and *plate* tensor voting. Since *plate* and *ball* tensor voting are not usually run at the same time, it is safe to avoid this new parameter.

TABLE I
SPEED MEASUREMENTS OF THE ORIGINAL TENSOR VOTING

Integration step (degrees)	*Plate* votes		*Ball* votes	
	Time (ms)	Votes per second	Time (s)	Votes per second
0.5	87.1	11.48	124.48	8.03×10^{-3}
1	79.3	12.61	41.10	2.43×10^{-2}
2	32.5	30.77	10.90	9.17×10^{-2}
5	19.0	52.63	1.75	0.57
10	12.0	83.33	0.45	2.22
20	6.6	152.88	0.12	8.01
30	5.6	178.25	0.05	19.19
45	5.3	189.04	0.03	28.49

V. EXPERIMENTAL RESULTS

The formulations of the original (OTV), efficient (ETV) and simplified tensor voting (STV) presented above were coded in MATLAB on an Intel Core 2 Quad Q6600 with a 4GB RAM in order to compare the new proposed schemes with the original tensor voting. In addition, the approximation scheme described in [4], [8], referred to as MM, has also been coded to compare its performance with the proposed methods. Equation (16) has been applied instead of (9) in order to make the results of all tested methods comparable.

A. Efficiency

Table I shows the mean execution times of the tested methods. This table shows that OTV is impractical for many applications. As an example, assume that a small cloud of points consists of 1,000 points, and that the propagation of votes is restricted to the 25 nearest points. Thus, the computation of 25,000 *stick*, *plate* and *ball* votes are required. With OTV, they can be calculated in between 16.85 minutes and 36.04 days depending on the desired precision (controlled by the integration step). For this reason, precomputing and storing the votes in voting fields by means of look-up tables is the only practical solution to apply OTV. Unfortunately, minimal or negligible loss of accuracy can only be attained through an expensive preprocessing stage to compute the voting fields using a small integration step. On the other hand, ETV does not requires preprocessing and only takes 0.05, 0.18, and 0.05 milliseconds for every *stick*, *plate* and *ball* vote respectively with an affordable loss of accuracy. Thus, in the aforementioned example, the proposed formulation only takes 7.05 seconds. In addition, this time can be further improved by implementing the method in a non-interpreted programming language, such as C/C++. The efficiency of STV is slightly better than ETV. In this case, the *stick*, *plate* and *ball* votes can be processed in 0.05, 0.15, and 0.04 milliseconds respectively on average. MM has also an efficient performance, since *plate* and *ball* votes can be processed in 0.20, and 0.04 milliseconds respectively on average. In these experiments, it is clear that the efficiency of OTV is affected by the use of loops in MATLAB. Although the relative improvement in speed from ETV, STV and MM is expected to decrease with a C/C++ implementation, such a reduction is rather limited since OTV is more computationally complex than ETV, STV and MM.

B. Comparisons with OTV

In order to assess the differences between OTV and the other methods, tensors computed with ETV, STV and MM have been compared to those obtained through OTV with a small integration step at a number of random points in the space. Two datasets of 1000 and 100 normally distributed random points with $\sigma = 1$ have been generated to assess *plate* and *ball* votes respectively. Fewer points have been tested for comparing *ball* votes in order to obtain results in a reasonable amount of time. For this experiment, OTV has been run with an integration step of 1 degree for $\sigma = 1$ and 5. Eight independent experiments have been run for $b = 0$, 1, 5 and 10 with $\sigma = 1$ and 5 respectively. Table II shows the differences between OTV and MM, ETV and STV. These differences have been computed through the mean angular error of $\mathbf{e_1}$ and $\mathbf{e_3}$ for *plate* votes cast by a *plate* at the origin ($\lambda_{1_\mathrm{P}} = \lambda_{2_\mathrm{P}} = 1$), and $\mathbf{e_3}$ for *ball* votes cast by a *ball* at the origin ($\lambda_{1_\mathrm{B}} = \lambda_{2_\mathrm{B}} = \lambda_{3_\mathrm{B}} = 1$), in addition to the root mean square error of the eigenvalues λ_1 and λ_2 for *plate* votes and λ_2 and λ_3 for *ball* votes. As shown in the table, ETV makes a better approximation of OTV than MM and STV. In particular, MM and STV introduce relevant diferences in eigenvalues for both *plate* and *ball* votes. This result was expected for STV since it does not aim at mimicking the behavior of OTV (cf. Section IV). It can also be seen that *ball* votes are equivalent for MM and STV with $\alpha_B = 0$, which was also expected. In summary, ETV is approximately equivalent to OTV, while MM and STV are different than OTV.

C. Accuracy

Accuracy has been measured by comparing the ground-truth with the results of applying the tested methods to some synthetic datasets. Figure 9 shows the point-sampled surfaces used in these experiments and their noisy counterparts. As suggested in [3], tensors were initialized with unitary *balls* and two rounds of tensor voting were applied: the first one only considered the *ball* tensor voting, whereas the second round only considered both the *stick* and *plate* tensor voting. Parameter α_B was set to zero, and σ was set to five. Independent experiments were run for $b = 0$ and $b = 10$. STV has been run with $\alpha_P = 0$ and $\alpha_P = 1$ in order to assess the effect of this parameter. For this and the following experiments, OTV has been computed by the interpolation of precomputed voting fields, since the application of OTV with a small integration step is impractical in this case.

The mean angular error between $\mathbf{e_1}$ and ideal normals on surfaces, and of $\mathbf{e_3}$ and ideal edge orientations at edges have been used to measure the accuracy of the algorithms for estimating normals and curve orientations respectively. In addition, the following measurement of discriminability [25] of saliencies has been used to assess the saliency estimation:

TABLE II
MEAN ANGULAR ERRORS OF EIGENVECTORS (IN DEGREES), AND ROOT MEAN SQUARE ERRORS OF THE EIGENVALUES

		Plate votes ($\sigma = 1$)				*Plate* votes ($\sigma = 5$)				*Ball* votes ($\sigma = 1$)			*Ball* votes ($\sigma = 5$)		
	b	$\mathbf{e}_1$	$\mathbf{e}_3$	λ_1	λ_2	$\mathbf{e}_1$	$\mathbf{e}_3$	λ_1	λ_2	$\mathbf{e}_3$	λ_2	λ_3	$\mathbf{e}_3$	λ_2	λ_3
MM	0	0.00	0.00	0.068	0.315	0.00	0.00	0.107	0.577	0.00	0.074	0.184	0.00	0.113	0.336
	1	0.00	0.00	0.116	0.311	0.00	0.00	0.195	0.513	0.00	0.120	0.125	0.00	0.216	0.251
	5	0.00	0.00	0.234	0.288	0.00	0.00	0.412	0.402	0.00	0.236	0.047	0.00	0.450	0.094
	10	0.00	0.00	0.300	0.246	0.00	0.00	0.534	0.322	0.00	0.299	0.020	0.00	0.575	0.039
ETV	0	0.00	0.00	0.040	0.015	0.00	0.00	0.072	0.030	0.00	0.009	0.008	0.00	0.013	0.013
	1	0.00	0.00	0.035	0.012	0.00	0.00	0.050	0.025	0.00	0.007	0.006	0.00	0.021	0.017
	5	0.00	0.00	0.019	0.012	0.00	0.00	0.028	0.022	0.00	0.013	0.013	0.00	0.031	0.029
	10	0.00	0.00	0.015	0.018	0.00	0.00	0.021	0.033	0.00	0.012	0.013	0.00	0.025	0.026
STV	0	0.00	0.00	0.382	0.140	0.00	0.00	0.702	0.264	0.00	0.074	0.184	0.00	0.113	0.336
	1	0.00	0.00	0.339	0.102	0.00	0.00	0.620	0.194	0.00	0.120	0.125	0.00	0.216	0.251
$\alpha_P = 0$	5	0.00	0.00	0.265	0.046	0.00	0.00	0.479	0.087	0.00	0.236	0.047	0.00	0.450	0.094
$\alpha_B = 0$	10	0.00	0.00	0.223	0.030	0.00	0.00	0.403	0.057	0.00	0.299	0.020	0.00	0.575	0.039
	0	0.00	0.00	0.068	0.140	0.00	0.00	0.107	0.264	0.00	0.564	0.330	0.00	1.069	0.620
STV	1	0.00	0.00	0.107	0.102	0.00	0.00	0.187	0.194	0.00	0.615	0.372	0.00	1.173	0.705
$\alpha_P = 1$	5	0.00	0.00	0.213	0.046	0.00	0.00	0.378	0.087	0.00	0.732	0.449	0.00	1.406	0.862
$\alpha_B = 1$	10	0.00	0.00	0.227	0.030	0.00	0.00	0.405	0.057	0.00	0.795	0.476	0.00	1.531	0.917

TABLE III
MEAN ANGULAR ERROR OF $\mathbf{e_1}$ AND $\mathbf{e_3}$ IN DEGREES AND DISCRIMINABILITY OF SALIENCIES FOR THE NOISELESS DATASETS OF FIG. 9

Method		Semisphere				Cone				Pyramid						
		Surf.		Curv.		Surf.		Curv.		Surf.		Curv.		Junctions		
	b	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	d_3	d_{3b}	d_{3t}
OTV	0	**0.33**	**0.08**	0.45	**0.05**	**0.90**	**0.09**	0.48	**0.09**	1.06	**0.16**	0.64	**0.15**	0.19	0.11	0.51
MM	0	**0.33**	**0.08**	0.56	0.04	**0.90**	0.08	0.58	0.08	1.12	0.15	0.74	0.14	0.19	**0.12**	0.46
ETV	0	**0.33**	**0.08**	0.45	**0.05**	**0.90**	**0.09**	0.48	**0.09**	1.08	**0.16**	0.65	**0.15**	0.19	0.11	0.50
STV ($\alpha_P = 0$)	0	**0.33**	**0.08**	**0.42**	**0.05**	**0.90**	**0.09**	**0.45**	**0.09**	1.09	**0.16**	0.68	**0.15**	0.18	**0.12**	0.41
STV ($\alpha_P = 1$)	0	**0.33**	**0.08**	0.44	**0.05**	0.91	**0.09**	0.51	**0.09**	**1.05**	**0.16**	**0.59**	**0.15**	**0.20**	0.11	**0.55**
OTV	10	**0.32**	**0.07**	0.32	**0.05**	**0.91**	0.08	0.35	**0.08**	1.17	**0.15**	0.89	0.13	0.19	**0.07**	0.68
MM	10	**0.32**	**0.07**	0.54	0.04	0.93	0.08	0.58	**0.08**	1.16	0.14	0.94	**0.14**	0.18	0.06	0.67
ETV	10	**0.32**	**0.07**	0.32	**0.05**	**0.91**	0.08	0.33	**0.08**	1.17	**0.15**	0.88	0.13	0.19	**0.07**	0.68
STV ($\alpha_P = 0$)	10	**0.32**	**0.07**	**0.31**	**0.05**	0.92	**0.08**	**0.30**	**0.08**	1.17	**0.15**	0.92	0.13	0.19	0.06	0.69
STV ($\alpha_P = 1$)	10	**0.32**	**0.07**	**0.31**	**0.05**	0.92	0.08	0.31	**0.08**	**1.15**	**0.15**	**0.83**	0.13	**0.21**	0.06	**0.80**

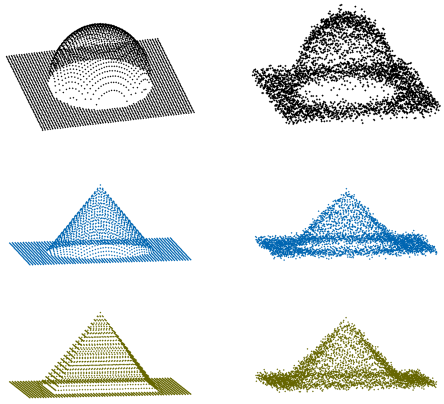

Fig. 9. Clouds of points used in the experiments. Left: point-sampled surfaces, each constituted by 3,721 points. Right: a noisy version of the same surfaces (Gaussian noise with standard deviation of 0.2).

$$d_1 = \frac{1}{||S||}\sum_{\mathbf{p}\in S}\frac{s_1(\mathbf{p})}{\lambda_1(\mathbf{p})} - \frac{1}{n-||S||}\sum_{\mathbf{p}\notin S}\frac{s_1(\mathbf{p})}{\lambda_1(\mathbf{p})} \quad (36)$$

$$d_2 = \frac{1}{||C||}\sum_{\mathbf{p}\in C}\frac{s_2(\mathbf{p})}{\lambda_1(\mathbf{p})} - \frac{1}{n-||C||}\sum_{\mathbf{p}\notin C}\frac{s_2(\mathbf{p})}{\lambda_1(\mathbf{p})} \quad (37)$$

$$d_3 = \frac{1}{||J||}\sum_{\mathbf{p}\in J}\frac{s_3(\mathbf{p})}{\lambda_1(\mathbf{p})} - \frac{1}{n-||J||}\sum_{\mathbf{p}\notin J}\frac{s_3(\mathbf{p})}{\lambda_1(\mathbf{p})} \quad (38)$$

where n is the total number of points in the dataset, S, C, and J are the set of points that belong to surfaces, curves and junctions respectively, and $||\cdot||$ is the cardinality of a set. In addition, d_3 has independently been computed for the junction at the top of the pyramid, d_{3t}, and for the junctions at the base, d_{3b}, since they represent two different types of junctions. As pointed out in [3], the classification of points into surfaces, curves and junctions cannot be performed by selecting the points where one saliency is larger than the others. Instead, the classification is performed by extracting local maxima of s_3 for junctions and through marching schemes for surfaces and curves, which also search for local maxima of s_1 and s_2 respectively. Thus, the proposed discriminability measurements estimate the degree of difficulty of deciding whether or not a point belongs to a surface, a curve or a junction respectively from the saliency measurements estimated through every method.

Tables III and IV summarize the results for the noiseless and noisy datasets respectively. Table III, shows that all methods yield similar angular errors in all datasets. The reason for this behavior is that the number of points that belong to surfaces, where *stick* votes are more important, is much higher than those that belong to curves or junctions. Thus, the total vote is much more influenced by the *stick* votes cast by neighboring points at the same surface than by *plate* votes cast by points located at neighboring curves. However, STV with $\alpha_P = 0$ has the better performance in curved datasets since it yields

smaller angular errors for $\mathbf{e_3}$. In turn, STV with $\alpha_P = 1$ has the best performance for the pyramid according to the mean angular errors. That means that points in curves are actually affected by *plate* votes cast from points located at neighboring surfaces. Errors related to *plate* votes are mitigated at points belonging to surfaces by the fact that they receive more *stick* votes from other points in the same surface. It is also important to note that parameter b barely influences angular errors and it tends to reduce the discriminability measurements in noiseless scenarios. Another observation with respect to this table is that discriminability measurements are relatively small. This does not suppose a problem for detecting curves and junctions, since they are located at points where saliencies s_2 and s_3 attain local maxima values respectively. Alternatively, iterative schemes can also be used to increase these discriminability measurements.

In turn, Table IV shows that, although the angular errors are higher, the observations made for noiseless scenarios are also valid in noisy ones. That is, STV yields the best results for curved scenarios by setting $\alpha_P = 0$ and for the pyramid by setting $\alpha_P = 1$. An interesting observation is that discriminability measurements d_1 and d_2 are similar to those reported in Table III. That means that the detection of curves is almost not influenced by noise. On the other hand, although the discriminability measurement d_{3t} is more affected by noise, the values are still high. In addition, the discriminability d_{3b} is less affected by noise. This means that junctions at the base of the pyramid can also be extracted from a noisy scenario.

Regarding MM, this experiment confirms the observation made in Table II in the sense that it is different from OTV, since both yield different results. In addition, the approximation made by MM usually yields worse results than OTV. However, it remains a good alternative to the methods proposed in this paper. As for ETV, this experiments show that it succeeds in mimicking the behavior of OTV, since it yields almost the result in all measurements.

D. Effect of Parameter α_P of STV

The effect of the *stick* component of *plate* votes has been assessed by measuring the distortion introduced by the methods when the tensors are initialized with the ground-truth for the cloud of points shown in Figure 10. The same measurements used in the previous experiment have been applied to this experiment and σ has been set to five. Table V shows that STV with $\alpha_P = 0$ is the method that less angular distortion introduces both for $b = 0$ and $b = 10$. However, the differences between STV and the other methods are reduced for $b = 10$. The reason of this behavior is that fewer points at curves are allowed to cast *stick* components, so their influence in the total vote is reduced in such a case. Although STV with $\alpha_P = 1$ is the method that induces less reductions in discriminability of saliencies, it has a bad performance in this dataset, since it introduces higher angular errors. That means that that setting $\alpha_P = 1$ is not appropriate for curved datasets. An expected result was a reduction in the discriminability of saliencies, which are one in the ground-truth, for all tested methods. Since these reductions are relatively small, especially for $b = 10$, they can be thought of as the small price that tensor voting has to pay for yielding robust results.

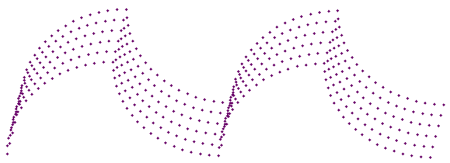

Fig. 10. Cloud of points used to assess the effect of the *stick* component of *plate* votes.

In addition, Table V shows the results on this dataset for tensors initialized with unitary *balls* and two applied rounds of tensor voting: one for *ball* votes and the other for *stick* and *plate* votes. For this dataset, angular errors and discriminabilities are larger than the values reported in Tables III and IV for other datasets. This is mainly due to two factors. First, the surfaces are intersected in an angle of 90 degrees, which maximizes the saliency s_2 in curves, leading to an increase in d_1 and d_2. Second, the dataset is rather sparse, so fewer votes are received at every point. Thus, the effect of an erroneous vote cannot be effectively compensated with enough correct ones, leading to an increase in the angular errors.

E. Effect of Parameter α_B of STV

Values of α_B greater than zero are only useful in iterative schemes where tensors have been initialized with bad estimations of the normals. In order to test the effect of this parameter, fifteen iterations of the *stick*, *plate* and *ball* tensor voting have been run for a sampled flat surface. Tensors have been initialized with one of the worst possible scenarios, that is, with tensors that are perpendicular to the normals in the surface. In particular, tensors have been initialized with *plate* tensors that are tangent to the surface. Thus, the angular error of $\mathbf{e}_1$ is 90 degrees at the beginning of the process. In addition, a small *ball* component has also been added to the tensors in order to force the application of the *ball* tensor voting in the first iteration. Parameter α_B has been set to ten for the first iteration and to zero for the subsequent iterations, since better estimations of the tensors are then available. Parameter σ has been set to two, while α_P has been set to zero. The arc cosine of the normalized tensor scalar product has been used to measure the tensor deviation, $tdev$, of the yielded tensor $\mathrm{T}(\mathbf{p})$ with respect to the ground-truth $\mathrm{T_g}(\mathbf{p})$ at every point of the dataset. This measure is given by [26], [27]:

$$tdev(\mathbf{p}) = \arccos\left(\frac{\langle \mathrm{T}(\mathbf{p}), \mathrm{T_g}(\mathbf{p})\rangle}{trace(\mathrm{T}(\mathbf{p}))\ trace(\mathrm{T_g}(\mathbf{p}))}\right), \quad (39)$$

where $\langle \mathrm{A}, \mathrm{B}\rangle = trace(\mathrm{A}\mathrm{B}^T)$ is the scalar product between tensors A and B. This measurement has the advantage that it is able to assess differences in orientation and anisotropy of the tensors at the same time.

Figure 11 shows the evolution with the iterations of the mean of $tdev$ and the mean of the saliency s_1 normalized by the largest eigenvalue λ_1. This figure shows that STV has

TABLE IV
MEAN ANGULAR ERROR OF $\mathbf{e_1}$ AND $\mathbf{e_3}$ IN DEGREES AND DISCRIMINABILITY OF SALIENCIES FOR THE NOISY DATASETS OF FIG. 9

Method		Semisphere				Cone				Pyramid						
		Surf.		Curv.		Surf.		Curv.		Surf.		Curv.		Junctions		
	b	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	d_3	d_{3b}	d_{3t}
OTV	0	5.62	0.07	3.90	**0.04**	5.62	**0.09**	5.06	**0.08**	5.21	0.15	4.30	**0.14**	**0.18**	0.11	0.44
MM	0	5.63	0.07	4.09	0.03	5.64	0.08	5.57	0.07	5.23	0.14	4.44	0.13	0.17	0.11	0.39
ETV	0	5.62	0.07	3.96	**0.04**	5.63	**0.09**	5.05	**0.08**	5.23	0.15	4.31	**0.14**	**0.18**	0.11	**0.45**
STV ($\alpha_P = 0$)	0	**5.61**	**0.08**	**3.68**	**0.04**	**5.59**	**0.09**	**4.85**	**0.08**	5.32	0.15	4.41	**0.14**	**0.18**	**0.12**	0.40
STV ($\alpha_P = 1$)	0	5.66	0.07	4.02	**0.04**	5.69	**0.09**	4.90	**0.08**	**5.18**	**0.16**	**4.28**	**0.14**	**0.18**	**0.12**	0.43
OTV	10	5.02	0.06	2.54	**0.06**	5.09	0.08	3.48	**0.08**	4.78	0.15	3.38	**0.13**	**0.15**	**0.08**	0.43
MM	10	5.09	0.06	3.20	0.04	5.12	0.08	4.24	**0.08**	4.78	0.14	3.70	**0.13**	**0.15**	**0.08**	0.43
ETV	10	5.01	0.06	2.53	**0.06**	5.09	0.08	3.46	**0.08**	4.79	0.15	3.37	**0.13**	**0.15**	**0.08**	0.44
STV ($\alpha_P = 0$)	10	**4.96**	**0.07**	**2.51**	**0.06**	**5.05**	**0.09**	**3.45**	**0.08**	4.81	0.14	3.50	**0.13**	**0.15**	0.07	0.46
STV ($\alpha_P = 1$)	10	4.98	0.06	2.60	**0.06**	5.09	0.09	3.68	**0.08**	**4.75**	**0.16**	**3.35**	**0.13**	**0.15**	0.07	**0.47**

TABLE V
MEAN ANGULAR ERROR OF $\mathbf{e_1}$ AND $\mathbf{e_3}$ IN DEGREES AND DISCRIMINABILITY OF SALIENCIES FOR THE CLOUD OF POINTS OF FIG. 10 FOR TWO TYPES OF INITIALIZATION

Method		Ground-truth				Unitary *balls*			
	b	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2	$\mathbf{e_1}$	d_1	$\mathbf{e_3}$	d_2
OTV	0	2.32	0.88	0.00	0.92	4.98	**0.72**	0.00	**0.59**
MM	0	2.07	0.82	0.00	0.92	5.02	0.66	0.00	0.57
ETV	0	2.30	0.88	0.00	0.92	4.96	0.71	0.00	**0.59**
STV ($\alpha_P = 0$)	0	**1.69**	0.87	0.00	0.90	**4.83**	0.70	0.00	0.57
STV ($\alpha_P = 1$)	0	2.84	**0.91**	0.00	**0.93**	4.98	**0.72**	0.00	0.58
OTV	10	1.87	0.96	0.00	**0.97**	4.70	**0.49**	0.00	0.39
MM	10	2.24	0.84	0.00	0.93	4.75	**0.49**	0.00	**0.40**
ETV	10	1.86	**0.97**	0.00	**0.97**	4.71	**0.49**	0.00	0.38
STV ($\alpha_P = 0$)	10	**1.56**	0.96	0.00	0.96	**4.59**	0.47	0.00	**0.40**
STV ($\alpha_P = 1$)	10	2.22	**0.97**	0.00	**0.97**	4.63	0.47	0.00	**0.40**

the best performance among the tested methods. In addition, α_B can be used to accelerate the convergence of STV. If fairly good initialization tensors are available, as it is the case from the second iteration onwards, α_B should be set to zero in order to avoid the introduction of unnecessary uncertainty. This experiment also shows the power of tensor voting in iterative schemes. Despite the poor initialization, all the implementations converge to the solution in a few iterations. In addition, it can also be seen in this experiment that the estimation of saliency s_1 tends to be improved with the number of iterations. Moreover, values of b greater than zero appear advantageous since all methods perform better in such a condition, especially for OTV and ETV. Furthermore, the experiment also shows that the performances of OTV and ETV are very close, while MM and STV perform differently.

VI. CONCLUDING REMARKS

This paper has proposed two alternative formulations in order to significantly reduce the high computational complexity of the *plate* and *ball* tensor voting. The first formulation makes numerical approximations of the votes, which have been derived from an in-depth analysis of the *plate* and *ball* voting processes. The second one proposes simplified equations to calculate votes that are based on the perceptual meaning of the original tensor voting. Both formulations have a complexity of order $\mathrm{O}(1)$. This can help broaden the use of tensor voting in more applications.

The numerical approach mimics the original formulation of tensor voting efficiently with a small error. On the other hand, the analytical approach has been found more appropriate for estimating saliencies at a cost of setting two new parameters. In both noisy scenarios and clouds of points with curved surfaces, the simplified tensor voting yields better results by setting the new parameter α_P to zero. In addition, higher values of α_P improve its performance for datasets with flat surfaces. Furthermore, parameter α_B can be used in iterative schemes where tensors are initialized with not too accurate values in order to artificially introduce uncertainty.

Moreover, perceptual interpretations for the *stick*, *plate* and *ball* tensor voting have been established. The *stick* tensor voting and the *stick* component of the *plate* tensor voting are used to reinforce surfaceness, whereas the *plate* components of both the *plate* and *ball* tensor voting are used to boost curveness. Junctionness is only intentionally strengthened by the *ball* component of the *ball* tensor voting.

Future work includes efficient implementations of the *plate* and *ball* votes in the frequency domain and extensions to higher dimensions.

APPENDIX

The functions s'_{1p} and s'_{2p}, required for computing *plate* votes in Section III have been divided into two parts, $0 \leq \gamma < \pi/4$ and $\pi/4 \leq \gamma < \pi/2$, in order to avoid the discontinuity at $\gamma = \pi/4$. These functions can be approximated through non-linear least squares fitting on γ as:

$$s'_{1p} \approx \begin{cases} \frac{c_{11}}{\gamma}\, e^{-\left(\frac{\ln(\gamma)-c_{12}}{c_{13}}\right)^2} & \text{if } 0 < \gamma \leq \pi/4 \\ 0 & \text{if } \gamma = 0 \\ c_{14} + c_{15}\, e^{-\left(\frac{\gamma-\pi/4}{c_{16}}\right)} & \text{otherwise,} \end{cases} \tag{40}$$

$$s'_{2p} \approx \begin{cases} e^{-\left(\frac{\gamma}{c_{21}}\right)^2} & \text{if } \gamma \leq \pi/4 \\ c_{22} + c_{23}\, e^{-\left(\frac{\gamma-\pi/4}{c_{24}}\right)} & \text{otherwise,} \end{cases} \tag{41}$$

where c_{1i} and c_{2j} for $i = 1, \ldots, 6$ and $j = 1, \ldots, 4$ are factors that must be computed only once, since they only depend on b. In turn, these factors can also be approximated through non-linear least squares on b by:

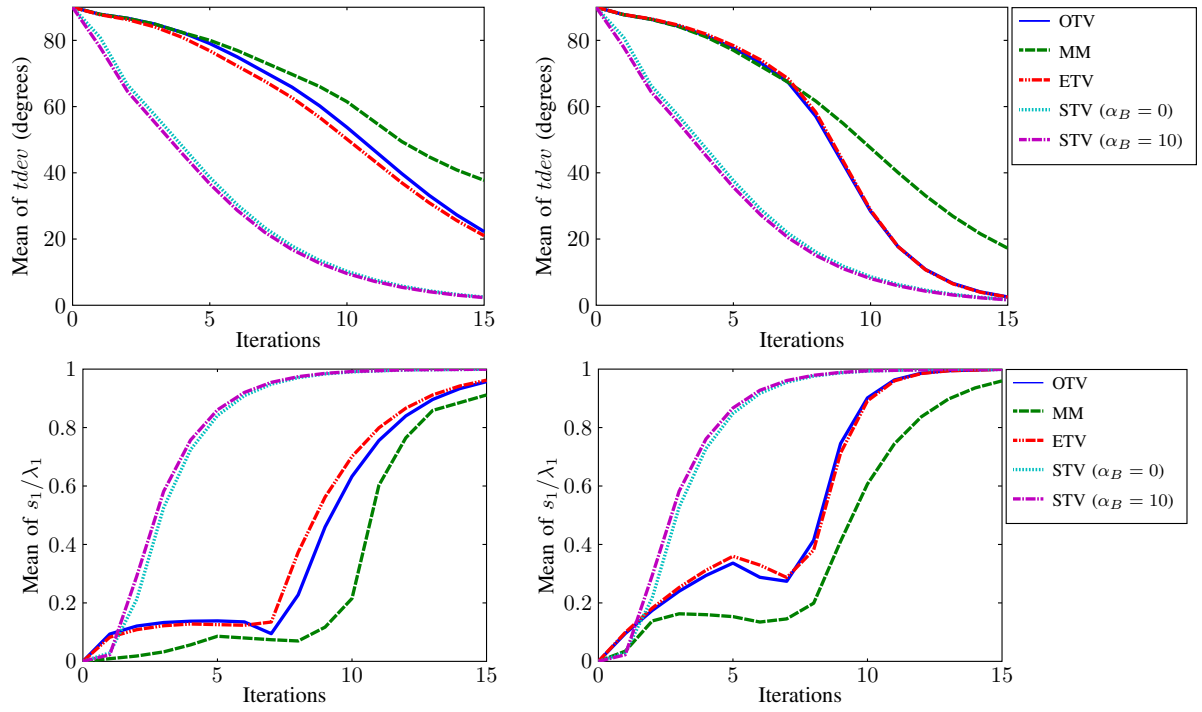

Fig. 11. Experiments on α_B for a flat surface with wrong initialization tensors. Top: evolution of the mean of $tdev$ with the iterations for $b = 0$ (left) and $b = 10$ (right). Bottom: evolution of s_1/λ_1 with the iterations for $b = 0$ (left) and $b = 10$ (right).

$$c_{11} \approx \frac{6.5360\ b - 3.7920}{b^2 + 4.8680\ b - 5.4220} - 0.3301\ e^{-0.1501\ b} + \frac{0.0600}{1 + e^{-0.0401\ (b-25.0)}} \quad (42)$$

$$c_{12} \approx 1.9501\ e^{-0.3010\ b} + 1.8001\ e^{-0.0101\ b} - 1.5750 \quad (43)$$

$$c_{13} \approx 1.2010 - 0.2511\ e^{-0.3001\ b} \quad (44)$$

$$c_{14} \approx 0.4901\ e^{-0.0801\ b} + 0.1301\ e^{-0.0010\ b} \quad (45)$$

$$c_{15} \approx 0.0785\ e^{-0.0221\ b} - 0.5701\ e^{-0.3501\ b} + 0.0466 \quad (46)$$

$$c_{16} \approx 0.1720\ e^{-0.2501\ (b-1.005)^2} + 0.0550 \quad (47)$$

$$c_{21} \approx 1.1197\ e^{-0.3114\ b} + 0.3552\ e^{-0.0125\ b} + 1.7894\ e^{-10.0\ b} + \frac{0.0450}{1 + e^{-(b-100.0)}} \quad (48)$$

$$c_{22} \approx 0.1977\ e^{-0.3101\ b} \quad (49)$$

$$c_{23} \approx 0.4278\ e^{-0.4051\ b} + 0.0836\ e^{-0.0881\ b} \quad (50)$$

$$c_{24} \approx 0.1294\ e^{-0.0878\ b} + 0.0243\ e^{0.0346\ b} \quad (51)$$

Following the same methodology, the functions s'_{2b} and s'_{3b}, required for computing *ball* votes, can be approximated by:

$$s'_{2b} \approx 0.2838\ e^{-0.0795b} + 0.2461\ e^{-0.0065b} \quad (52)$$

$$s'_{3b} \approx 0.3380\ e^{-0.3197b}. \quad (53)$$

In addition, s'_{3b} can be approximated by s'_{2b}/b for high values of b (*e.g.*, $b > 30$).

REFERENCES

[1] G. Guy and G. Medioni, "Inferring global perceptual contours from local features," *Int. J. Comput. Vis.*, vol. 20, no. 1-2, pp. 113–133, 1996.

[2] ——, "Inference of surfaces, 3D curves and junctions from sparse, noisy 3D data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 11, pp. 1265–1277, 1997.

[3] G. Medioni, M. S. Lee, and C. K. Tang, *A Computational Framework for Feature Extraction and Segmentation*. Elsevier Science, 2000.

[4] P. Mordohai and G. Medioni, *Tensor Voting: A perceptual Organization Approach to Computer Vision and Machine Learning*. Morgan and Claypool Publishers, 2006.

[5] V. Bruce, P. R. Green, and M. A. Georgeson, *Visual Perception: physiology, psychology and ecology*, 4th ed. Psychology Press, 2003.

[6] E. Franken, M. van Almsick, P. Rongen, L. Florack, and B. ter Haar Romeny, "An efficient method for tensor voting using steerable filters," in *Proc. ECCV*, 2006, pp. IV:228–240.

[7] M. Reisert and H. Burkhardt, "Efficient tensor voting with 3D tensorial harmonics," in *Proc. CVPR*, 2008, pp. 1–7.

[8] P. Mordohai and G. Medioni, "Dimensionality estimation, manifold learning and function approximation using tensor voting," *J. of Mach. Learn.*, vol. 11, pp. 411–450, 2010.

[9] D. Lu, H. Zhao, M. Jiang, S. Zhou, and T. Zhou, "A surface reconstruction method for highly noisy point clouds," in *Proc. Var. Geom. and Lev. Set Methods in Comput. Vis.*, 2005, pp. 283–294.

[10] C. Min and G. Medioni, "Tensor voting accelerated by graphics processing units (GPU)," in *Proc. ICPR*, 2006, pp. III:1103–1106.

[11] T.-P. Wu, J. Jia, and C.-K. Tang, "A closed-form solution to tensor voting for robust parameter estimation via expectation-maximization," Hong Kong Univ. Sci. Technol., Tech. Rep., 2009.

[12] T.-P. Wu, S.-K. Yeung, J. Jia, and C.-K. Tang, "Quasi-dense 3D reconstruction using tensor-based multiview stereo," in *CVPR*, 2010, pp. 1482–1489.

[13] D. Sornette, *Critical Phenomena in Natural Sciences. Chaos, Fractals, Selforganization and Disorder: Concepts and Tools*, 2nd ed. Springer Berlin Heidelberg, 2006.

[14] T. Marchant, "Scale invariance and similar invariance conditions for bankruptcy problems," *Soc. Choice Welf.*, vol. 31, no. 4, pp. 693–707, 2008.

[15] K. Belbahri, "Scale invariant operators and combinatorial expansions," *Adv. Appl. Math.*, vol. 45, no. 4, pp. 548–563, 2010.

[16] L. Loss, G. Bebis, M. Nicolescu, and A. Skurikhin, "An iterative multiscale tensor voting scheme for perceptual grouping of natural shapes in cluttered backgrounds," *Comput. Vis. and Image Underst.*, vol. 113, no. 1, pp. 126–149, 2009.

[17] W. S. Tong, C. K. Tang, P. Mordohai, and G. Medioni, "First order augmentation to tensor voting for boundary inference and multiscale analysis in 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 5, pp. 594–611, 2004.

[18] P. W. Bridgman, *Dimensional analysis*. Yale University Press, 1922.

[19] E. W. Andrews and L. J. Gibson, "The role of cellular structure in creep of two-dimensional cellular solids," *Mater. Sci. Eng. A*, vol. 303, no. 1-2, pp. 120–126, 2001.

[20] J. T. Ricketts, M. K. Loftin, and F. S. Merritt, *Standard Handbook for Civil Engineers*, 5th ed. McGraw-Hill Professional, 2003.

[21] F. Kreith and D. Y. Goswami, Eds., *The CRC handbook of Mechanical engineering*, 2nd ed. CRC Press, 2005.

[22] D. J. Magee, *Orthopedic Physical Assessment*, 5th ed. Saunders, 2008.

[23] Z. Miller and M. B. Fuchs, "Effect of trabecular curvature on the stiffness of trabecular bone." *J. Biomech.*, vol. 38, no. 9, pp. 1855–1864, 2005.

[24] S. Fischer, P. Bayerl, H. Neumann, G. Cristobal, and R. Redondo, "Iterated tensor voting and curvature improvement," *Signal Process.*, vol. 87, no. 11, pp. 2503–2515, 2007.

[25] R. Moreno, D. Puig, C. Julià, and M. A. Garcia, "A new methodology for evaluation of edge detectors," in *Proc. ICIP*, 2009, pp. 2157–2160.

[26] T. Peeters, P. Rodrigues, A. Vilanova, and B. ter Haar Romeny, *Visualization and Processing of Tensor Fields: Advances and Perspectives*. Springer, Berlin, 2009, ch. Analysis of distance/similarity measures for Diffusion Tensor Imaging, pp. 113–138.

[27] L. Jonasson, X. Bresson, P. Hagmann, O. Cuisenaire, R. Meuli, and J. Thiran, "White matter fiber tract segmentation in DT-MRI using geometric flows," *Med. Image Anal.*, vol. 9, pp. 223–236, 2005.

Rodrigo Moreno Rodrigo Moreno received B.S. and M.S. degrees in computer science from the University of Los Andes (Bogotá, Colombia) in 1995 and 1997 respectively; a B.S. degree in electrical engineering from the National University of Colombia (Bogotá, Colombia) in 1995; a M.S. degree in organizations management from the University of Quèbec (Chicoutimi, Canada) in 2006; and a Diploma in Advanced Studies and his Ph.D. degree from the Polytechnic University of Catalonia (Barcelona, Spain) in 2007 and 2010 respectively. In 1999, he joined St. Martin University (Bogotá, Colombia), where he was the Head of the Department of Computer Science from 2001 to 2006. In 2006, he joined the Intelligent Robotics and Computer Vision Group at Rovira I Virgili University (Tarragona, Spain). He is currently a Post-Doctoral Researcher at the Center for Medical Image Science and Visualization and the Deptartment of Medical and Health Sciences at Linköping University (Linköping, Sweden). His research interests include image analysis, computer vision, tensorial methods and biomedical imaging.

Miguel Angel Garcia Miguel Angel Garcia received the B.S., M.S., and Ph.D. degrees in computer science from the Polytechnic University of Catalonia (Barcelona, Spain) in 1989, 1991, and 1996 respectively. He joined the Department of Software at the Polytechnic University of Catalonia in 1996 as an Assistant Professor. From 1997 to 2006, he was with the Department of Computer Science and Mathematics at Rovira i Virgili University (Tarragona, Spain), where he was the Head of Intelligent Robotics and Computer Vision group. In 2006, he joined the Department of Informatics Engineering at Autonomous University of Madrid (Spain), and in 2010 the Department of Electronic and Communications Technology, at Autonomous University of Madrid (Spain), where he is currently Associate Professor and member of the Video Processing and Understanding Lab. His research interests include image processing, computer vision, 3-D modeling, and mobile robotics.

Domenec Puig Domenec Puig received the M.S. and Ph.D. degrees in computer science from the Polytechnic University of Catalonia (Barcelona, Spain), in 1992 and 2004 respectively. In 1992, he joined the Department of Computer Science and Mathematics at Rovira i Virgili University (Tarragona, Spain), where he is currently Associate Professor. Since July 2006, he is the Head of the Intelligent Robotics and Computer Vision group at the same university. His research interests include image processing, texture analysis, perceptual models for image analysis, scene analysis, and mobile robotics.

Luis Pizarro Luis Pizarro received the B.E. and M.S. degrees in Informatics Engineering from the Technical University Federico Santa Maria (Chile) in 2003. From 2001 to 2003 he worked as a research assistant at the same university and was a consulting engineer at Anglo American Chile S.A. In 2004 he spent five months as a research trainee at the INRIA Sophia-Antipolis, France. From 2004 to 2005 he worked as a researcher at the Mining and Metallurgy Innovation Institute, IM2, (Chile). From 2006 to 2010 he pursued the Ph.D. degree in Computer Science at Saarland University (Saarbrücken, Germany). Since October 2009 he is with the Department of Computing and the National Heart and Lung Institute at Imperial College (London, UK), as a Post-Doctoral Researcher. His research interests include image analysis, computer vision, machine learning and biomedical imaging.

Bernhard Burgeth Bernhard Burgeth received his diploma and doctoral degree in mathematics from the University of Erlangen-Nürnberg (Germany) in 1991 and 1996, respectively. He worked as research assistant at the University of Erlangen-Nürnberg, as researcher (DFG research grant) at McGill University, Montreal (Canada), at the Technical University Eindhoven (The Netherlands), at the Karlsruhe Research Center (Karlsruhe, Germany), and, as a assistant professor in the Mathematical Image Analysis Group at Saarland University (Saarbrücken, Germany). After his habilitation in Mathematics in 2009, he joined the Faculty of Mathematics and Computer Science at Saarland University (Saarbrücken, Germany) as an associate professor. His research interests include partial differential equations and their applications to the processing of tensor fields and medical image analysis in general.

Joachim Weickert Joachim Weickert is professor of Mathematics and Computer Science at Saarland University (Saarbrücken, Germany), where he heads the Mathematical Image Analysis Group. He graduated and obtained his Ph.D. from the University of Kaiserslautern (Germany) in 1991 and 1996. He worked as Post-Doctoral Researcher at the University Hospital of Utrecht (The Netherlands) and the University of Copenhagen (Denmark), and as Assistant Professor at the University of Mannheim (Germany). Joachim Weickert has developed many models and efficient algorithms for image processing and computer vision using partial differential equations and optimisation principles. In particular he has contributed to diffusion filtering, optical flow computation, processing of tensor fields, and image compression. His scientific work covers about 230 refereed publications. He has served in the editorial boards of nine international journals and given more than 130 invited talks. In 2010 he has received the Gottfried Wilhelm Leibniz Prize which is the highest German science award.