Robust Estimation of Adaptive Tensors of Curvature by Tensor Voting

Wai-Shun Tong, *Student Member*, *IEEE*, and Chi-Keung Tang, *Member*, *IEEE*

Abstract—Although curvature estimation from a given mesh or regularly sampled point set is a well-studied problem, it is still challenging when the input consists of a cloud of unstructured points corrupted by misalignment error and outlier noise. Such input is ubiquitous in computer vision. In this paper, we propose a three-pass tensor voting algorithm to robustly estimate curvature tensors, from which accurate principal curvatures and directions can be calculated. Our quantitative estimation is an improvement over the previous two-pass algorithm, where only qualitative curvature estimation (sign of Gaussian curvature) is performed. To overcome misalignment errors, our improved method automatically corrects input point locations at subvoxel precision, which also rejects outliers that are uncorrectable. To adapt to different scales locally, we define the $RadiusHit$ of a curvature tensor to quantify estimation accuracy and applicability. Our curvature estimation algorithm has been proven with detailed quantitative experiments, performing better in a variety of standard error metrics (percentage error in curvature magnitudes, absolute angle difference in curvature direction) in the presence of a large amount of misalignment noise.

Index Terms—Curvature, curvature tensor, tensor voting.

1 Introduction

Curvature information has been well-known for its importance, usefulness, and elusiveness for accurate estimation. While approaches based on second order derivatives, surface fitting, scatter matrices, and mesh-based methods have made some significant progress in its quantitative estimation, the robustness to quantization errors and outlier noise is still an issue. Some approaches resort to qualitative estimation (e.g., sign of Gaussian curvature) or require a mesh as input.

The *tensor of curvature* has recently attracted some attention [16]. It has been shown as a promising alternative for the robust estimation of principal curvatures and directions. But most previous curvature tensor estimation approaches [15], [23] assume the presence of a mesh or require input points be sampled in structured grids. Preprocessing to obtain a mesh and noise robustness are therefore issues, especially in the presence of multiple feature scales.

In this paper, we propose a tensor voting approach to estimate tensors of curvature that adapts to different scales, given a noisy and unorganized input point set. In *three* tensor voting passes, we estimate normal direction information for all input tokens or points and reject outliers that are not lying on any smooth surface (pass 1). For correctable outliers and identified inliers, our method automatically corrects their positions to eliminate quantization errors (pass 2) and estimates curvature tensors at the corrected positions (pass 3). Points in structured grids (such as MRI) is not required. Our method does not rely on a mesh as input, which may prematurely limits the estimation precision.

• *The authors are with Vision and Graphics Group, Computer Science Department, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: {cktang, cstws}@cs.ust.hk.*

Manuscript received 18 July 2003; revised 30 July 2004; accepted 23 Aug. 2004; published online 14 Jan. 2005.
Recommended for acceptance by S. Soatto.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0181-0703.

While in all three passes of tensor voting, multiple scales of analysis are used, recent development on multiscale tensor voting is addressed elsewhere [24]. Adaptive scaling in curvature tensor estimation (pass 3) is specifically addressed in this curvature paper.

To position our curvature estimation approach, we directly compare our method with related curvature tensor approaches: Taubin [23], Tang and Medioni [20], [21], and Page et al. [15]. These three approaches do not compute second order partial derivatives, as adopted by many other methods. Instead, they estimate the tensor of curvature (or its variants). Our work represents a significant advancement over a previous approach [20], [21], which estimates signs of Gaussian curvatures only and assumes a single scale of analysis. Similar to [20], our method does not estimate partial second order derivatives (which is very error-prone), does not perform explicit local surface fitting (which may commit wrong decision early), and does not require initial oriented normal estimation (which requires the difficult maintenance of normal consistency on an unknown surface). However, in [20], [21], a two-pass tensor voting algorithm is proposed for estimating sign of Gaussian curvature only. In this paper, a three-pass algorithm is proposed in order to accurately estimate curvature signs, magnitudes, and directions in the presence of noise and multiple feature scales. If we need to output a surface, curvature-based voting fields can be used to vote for surfaces [20], [21].

Detailed experiments have been performed to confirm our findings. Qualitative and quantitative tests on both synthetic and real data in the presence of large amount of outlier noise *and* quantization/misalignment noise were performed, which show the robustness and graceful degradation of our tensor voting approach for severely contaminated data.

The organization of this paper is as follows: Section 2 reviews previous work. Section 3 gives a contextual review of tensor voting for curvature tensor estimation. Section 4 reviews the tensor voting algorithms. Section 5 describes our tensor voting approach for curvature tensor estimation

in detail. Section 6 quantifies the estimation accuracy and applicability of our method. Quantitative and qualitative results are shown and explained in Section 7. Finally, we conclude our paper in Section 8.

2 Related Work

Our work on principal curvature and direction estimation is grounded on differential geometry (see the standard text by do Carmo [3]). Many papers on surface curvature estimation from range data were published in the 1980s. Among these classical studies, Flynn and Jain [7] performed empirical studies on five methods of curvature estimation available at that time and concluded experimentally that qualitative curvature measures (e.g., signs of Gaussian curvatures [1]) can be reliably estimated. However, quantitative properties, such as the magnitudes of curvatures, are difficult to estimate robustly in the presence of noise. Trucco and Fisher [26] had a similar conclusion.

In most cases, the goal is to estimate all the local surface geometry, or the Darboux frames proposed by Petitjean [16] and Sander and Zucker [19], though it cannot always be accurately estimated. The Darboux frames can be defined as:

$$\Delta P = (P, n, p_1, p_2, k_1, k_2), \tag{1}$$

where P is a point on the surface, n is the surface normal at that location, and p_1 and p_2 are the principal curvature directions at which the maximum curvature, k_1, and minimum curvature, k_2, occur. On the other hand, the mean curvature and Gaussian curvature are defined as $\frac{(k_1+k_2)}{2}$ and k_1k_2, respectively.

2.1 Local Surface Fitting

It is known that directly computing second order derivatives is very sensitive to noise and quantization errors. Local surface or model fitting is therefore commonly used. The surface is often assumed to be locally planar, biquadratic, bicubic, or biquartic. Besl and Jain [2] used surface fitting from the lowest order until model misfit occured and then increased the order of complicity. Quadric patches are also very popular, as in [2], [4], [6], [18], [27]. Petitjean [16] provided a comprehensive survey on recovering of quadrics from triangle meshes. We will briefly discuss some of the methods below. In a classical and important work by Sander and Zucker [19], a comprehensive description on local surface fitting was provided, where iterative refinement of local surface estimation is based on local neighborhood support for extracting a local surface parameterization. In robotics, Carlebois et al. [8] used a B-spline to perform local fitting. In fact, no matter what surface parameterization is used, as long as the partial derivatives are computed from the model, the Darboux frames can be found.

2.2 Direct Differential Geometry Approach

When accuracy can be relaxed and efficiency is of primary concern, direct computation of local differential geometry without fitting a surface can be performed. For $2\frac{1}{2}$D input, only local gradient information is used in [17]. It is also possible to estimate curvature from structured grid points, such as MRI, directly without surface fitting. On the other hand, Martin [13] approximated local curves by fitting circles. The algorithm works by choosing triples of points in the immediate neighborhood of a point P to form circles C_j. Multiple tangents of the surface can then be obtained at P. Usually, a set of tangents define a tangent plane at P. The normal can also be estimated by the cross product of any two tangents.

We let t_0 be an arbitrary reference direction at P and θ_0 be the angle between the first principal direction and t_0. By using the Euler formula, we have a set of equations: $k_n(t'_j) = k_1\cos^2(\theta_j - \theta_0) + k_2\sin^2(\theta_j - \theta_0)$, where θ_j is the angle between each of the t'_j and t_0. By solving the set of equations using the least square method, the Darboux frame can be found.

In [10], the estimation of directional curvature (i.e., the curvature of a curve on the surface on a given point and at a given tangent direction) is given by: $c_{ij} \approx \pm\frac{\angle p_i p_{ij} p_j}{d_{ij}} \approx \pm\frac{\cos^{-1}(n_i, n_j)}{d_{ij}}$, where p_i and p_j are two neighboring vertices with normal estimated as n_i and n_j. p_{ij} is the center of the circle going through p_i, p_j, and d_{ij} is the distance between them. k_1 and k_2 can then be estimated by taking the maximum extremum and minimum extremum among the c_{ij}. Direct differential geometry methods can guarantee accuracy when the data is of high resolution and adequately accurate.

2.3 Multiscale Curvature

As curvature estimation is highly sensitive to noise, some researchers have proposed obtaining a more stable curvature estimation by means of smoothing or using a multiscale local surface description. While some only performed smoothing on the mesh, others modified their parameterization so that smoothing can be done within their formulation. In [29], multiscale curvature computation on a smoothed 3D surface was proposed. Semigeodesic coordinates are constructed at each vertex space. The Gaussian and mean curvatures are then estimated using a traditional curvature estimation technique. While their experiments showed that curvature estimation is more stable after smoothing, smoothing inevitably results in overestimation (respectively, underestimation) of minimum (respectively, maximum) curvatures. In [5], Dudek and Tsotsos also used a multiscale curvature technique to make their shape representation and recognition system more stable.

2.4 Robotic Approach in Curvature Sensing

In [8], Charlebois et al. investigated the use of robotic probes for finding curvature information of a surface. EP1 (probe with two points of contact) and EP2 (probe with finger-like contact) were used. It was concluded that EP1 was not accurate. EP2 was more accurate when a B-Spline surface is fitted by using the data obtained from the probe. It is very interesting that the authors had a similar finding for the EP1 sensitivity, as compared with our $RadiusHit$ of a curvature tensor for directional curvature estimation. In their experiment, if the resolution of the sensor was not high, they needed a larger distance of measurement. This observation is analogous to a large $RadiusHit$ value to improve the estimation. However, the improvement is up to a certain point. This is related to the choice of neighborhood size, or the local scale of analysis.

2.5 Evaluation on Curvature Estimation

We summarize one of the findings from Trucco and Fisher [26] here concerning the use of curvature information for segmentation. They compared several methods for curvature based segmentation for range data. They found that the accuracy of the mean curvature is largely limited by the

TABLE 1
Comparison of Parameters Used for Estimating Curvature Tensor Defined by (2)

	Taubin	Tang & Medioni	Page et al.	this paper
Input	mesh	points or mesh	mesh	points or mesh
N_p	weighted area average	inferred by tensor voting	normal voting with weighted area	inferred by tensor voting
t_{ij}	tangent line	tangent line	tangent line	tangent line
k_{ij}	$\frac{2\cos\frac{\pi-\varphi}{2}}{\|\|r-c\|\|}$ (Fig. 5)	$\frac{2\cos\frac{\pi-\varphi}{2}}{\|\|r-c\|\|}$ (Fig. 5)	$\frac{\varphi}{\text{geodesic distance}}$	$\frac{2\cos\frac{\pi-\varphi}{2}}{\|\|r-c\|\|}$ (Fig. 5)
w_{ij}	triangle area	vote saliency $\propto$ distance	triangle area	$\frac{1}{\text{no. of samples}}$ i.e. $(\frac{1}{8})$
Neighborhood	adjacent triangular vertices	defined by σ	defined by σ	$45°$ apart on the tangent plane within *RadiusHit*
Output	Darboux frames	Signs and directions	Darboux frames	Darboux frames
Scale selection	free parameter	free parameter	free parameter	adaptive and automatic
Accuracy metric	none	none	none	*RadiusHit* of curvature tensor

combination of quantization and smoothing error. While smoothing is necessary for better estimation of curvature for quantized data, it will lower the actual curvature value. In one of their test cases for sphere curvature estimation, they found that decreasing the radii of their sphere data set also decreases the accuracy. As smoothing a small radius sphere will flatten it, the estimated curvature values will be lower. They concluded that using a plane fitting method, as proposed by [28] to first segment out the planes, can allow for a better classification of the remaining curves.

2.6 Comparison of Curvature Tensor Methods

We summarize the comparison among four methods (including the one in this paper) on estimating curvature tensor in Table 1 and provide some details in the following sections. To facilitate our comparison, all notations in the rest of this section are from [23].

2.6.1 Discrete Estimation by Taubin

In differential geometry, it is known that a surface can be reconstructed up to second order (except for a constant term) given the two principal curvatures at each point, using the first and second fundamental forms. Therefore, curvature information provides a useful shape descriptor for various tasks in computer vision, ranging from image segmentation and feature extraction to scene analysis and object recognition.

If a mesh is available, Taubin [23] proposed estimating the curvature tensor at a mesh vertex, v_i, by computing the following weighted tensor sum over the neighborhood $nbhd(v_i)$:

$$M_{v_i} = \sum_{v_j \in nbhd(v_i)} w_{ij} k_{ij} t_{ij} t_{ij}^T , \quad (2)$$

where t_{ij} is the unit vector resulted by the projection of $v_j - v_i$ onto the tangent plane at v_i (i.e., t in Fig. 5). The normal curvature k_{ij} is estimated by the formula in Table 1. The weight w_{ij} is proportional to the sum of the areas of all triangles incident to v_i and v_j. Note that $t_{ij}t_{ij}^T$ is a second order symmetric tensor represented by a symmetric matrix. Note that a similar formula also appears in [12].

2.6.2 Discrete Estimation by Tang and Medioni

The v_i in (2) becomes a vote receiver (votee) and the set of $v_j \in nbhd(v_i)$ are voters in the tensor voting formulation for robust curvature estimation proposed by Tang and Medioni [20]. This is a two-pass algorithm: normal and curvature estimation by tensor voting. A second order symmetric tensor in 2D is voted for, which lies on the tangent plane at v_i. t_{ij} is derived from the 2D stick voting field [14]. Each t_{ij} contributes a stick tensor $t_{ij}t_{ij}^T$, which is accumulated by tensor addition. Similar to Taubin [23], k_{ij} here is the curvature of the hypothesized circular arc connecting v_i and v_j. w_{ij} is the vote consistency, given by the inner product $< N_{v_j}, N_{ij} >$, where N_{v_j} is the estimated normal at v_j and N_{ij} is the normal vote induced at v_j. Since only signs of Gaussian curvature are estimated in [20], [21], s_{ij} is not necessarily proportional to w_{ij}. The estimation is robust to quantization noise because only signs are estimated. But, without subvoxel position correction, it will produce unacceptable results if magnitudes of principal curvatures are estimated.

2.6.3 Discrete Estimation by Page et al.

Inspired by [23] and [20], Page et al. estimated principal curvatures and principal directions from a triangular mesh. They proposed a voting method called normal voting, which is similar to [20], [21], to first vote for the normal direction at v_i. Then, curvature votes along directions t_{ij} around v_i were collected into a curvature tensor M_{v_i}, given by (2). A mesh input is needed.

In [15], k_{ij} is defined as the ratio of the geodesic distance to the change in turning angle. We implemented this alternative of computing curvature and tested it on synthetic data, for which ground truth is available. (Note that the other two methods [20], [21], [23] use the definition of curvature in differential geometry, $\frac{2\cos\frac{\pi-\varphi}{2}}{\|r-c\|}$, or simply the reciprocal of the radius of curvature). We found the principal curvatures given by the geodesic distance to turning angle ratio is only incrementally better when the two points are farther apart (in this case, it will not produce a more accurate result anyway). To save computation time, in this work, we still use the closed form adopted in [20], [21], [23].

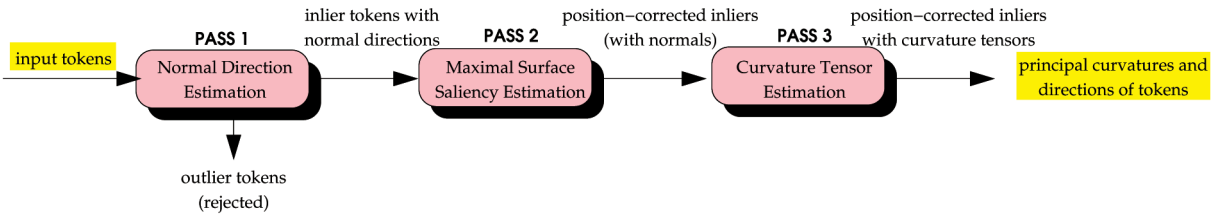

Fig. 1. Curvature tensor estimation by tensor voting.

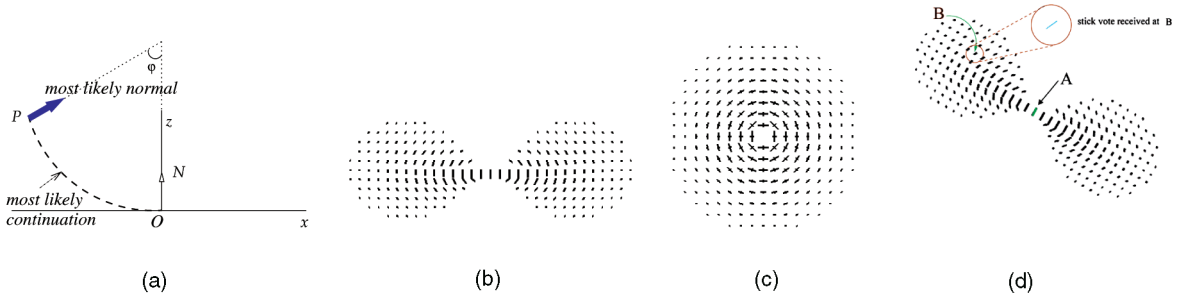

Fig. 2. (a) Design of the stick voting field. One slice of the two voting fields in 3D: (b) stick voting field, (c) ball voting field, and (d) 2D illustration on how tensor voting is performed. Here, B receives a stick vote from A after field alignment.

In Page et al. [15], the weight function w_{ij} relates to the surface area of triangles, the geodesic neighborhood, and a scale parameter.

2.7 Sampling

In a classical study [6], it is proven that computing the principal curvatures and directions is equivalent to computing curvature in four different directions 45 degrees apart. In our experiments, we found that *uniform sampling* of directional curvatures at a fixed stepping angle on the tangent plane is crucial to the accuracy of the estimated tensor when discrete integration is performed to accumulate a curvature tensor. This is implicitly addressed by the weighted area constraint in [15], [23]. Using these results, we estimate (vote for) directional curvatures, each differs by 45 degrees, and use them to estimate the curvature tensor. The use of more than four directional curvatures only has an incremental improvement on the estimation accuracy, while more processing time is needed.

3 Contextual Review of Tensor Voting

In this section, we give a concise review of tensor voting [14] in the context of our three-pass algorithm, where *normal tensor votes* and *curvature tensor votes* are sampled. The flowchart is shown in Fig. 1, where the new algorithm (Algorithm 5) in curvature tensor estimation in pass 3 compactly summarizes the procedure.

3.1 Token Encoding

Initially, since no normal direction is available, each input point is encoded as a ball tensor token, represented by the equivalent eigensystem with eigenvalues $\lambda_1 = \lambda_2 = \lambda_3 = 1$ and eigenvectors $\hat{e}_1, \hat{e}_2, \hat{e}_3$, which are arbitrary orthonormal vectors.

3.2 Token communication

For each point token P, the tokens in P's neighborhood cast tensor votes to P by using the *3D ball voting field* if no direction information is available. If the normal direction is available, a *3D stick voting field* is used. The 3D ball voting field can be derived from the 3D stick voting field.

The design of the stick voting field is shown in Fig. 2a. Suppose there exists a smooth curve connecting the origin O and a point P. Suppose also that the normal N to the curve at O is known. What is the most likely normal direction at P? Note that O, P, and N lie on the same plane. The osculating circle connecting O and P is chosen as the most likely connection since it keeps the curvature constant along the hypothesized circular arc. The most likely normal is given by the normal to the circular arc at P (thick arrow in Fig. 2a). The length of this normal, which represents the vote saliency, is inversely proportional to the arc length OP and also to the curvature of the underlying circular arc. The decay of the field is defined as:

$$\overline{DF}(r, \kappa, \sigma) = e^{-\left(\frac{r^2 + C\kappa^2}{\sigma^2}\right)}, \tag{3}$$

where r is the arc length OP, $\kappa = 2\cos\varphi / ||O - P||$ is the curvature, C is a constant which controls the decay with high curvature, and σ determines the effective neighborhood size. The derivation of the field decay function is described in detail in [14], where a study is given on how tokens are connected in the presence/absence of normal directions.

If we consider all points in the 3D space, the whole set of normals derived constitutes the 3D stick voting field, describing the direction $[0\ 0\ 1]^T$ and postulating directions in a neighborhood. The 3D ball voting field is generated by rotating the 3D stick voting field about the x-axis and y-axis, integrating the vote contributions by tensor summation. Fig. 2b and Fig. 2c show one slice of each voting field, where the different size of each tensor element (called stick and ball vote respectively) indicates the vote saliency.

A point A *casts a stick vote* to B by aligning the stick voting field with A's normal, if it is present. Otherwise, A casts a ball vote to B using the ball voting field, where no alignment is needed for the isotropic field. In either case, the size of the voting field is determined by σ, Fig. 2d. The votes received at

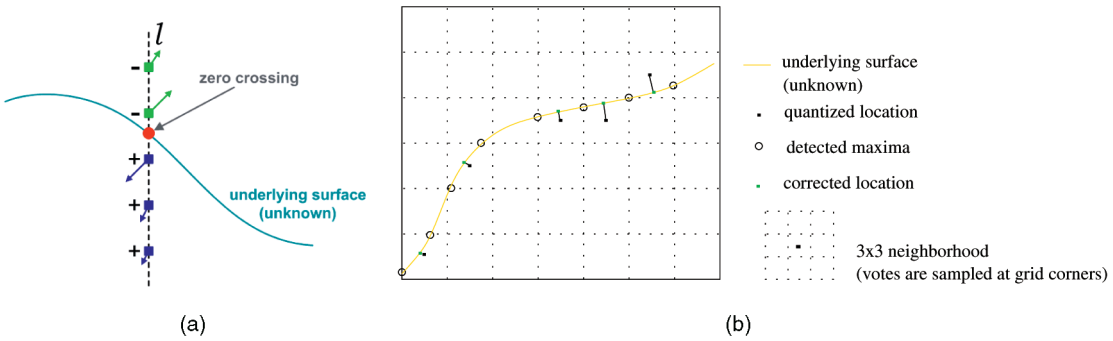

Fig. 3. (a) Illustration of zero crossing detection. (b) A 2D illustration of token position correction by tensor voting when maximal surface saliency is computed (pass 2 in Fig. 1).

B are accumulated by summing up the second order moment of the collected votes into the covariance matrix. For example, let B receive a total of m stick votes in its neighborhood after voting with the 3D stick voting field. Denote a stick vote by $[v_x v_y v_z]^T$. The *normal tensor vote* at P is given by

$$\mathbf{S} = \begin{bmatrix} \sum_m v_x^2 & \sum_m v_x v_y & \sum_m v_y v_z \\ \sum_m v_y v_x & \sum_m v_y^2 & \sum_m v_x v_z \\ \sum_m v_y v_z & \sum_m v_x v_z & \sum_m v_z^2 \end{bmatrix}.$$

3.3 Vote Interpretation

$\mathbf{S}$ is a symmetric and semipositive definite, which can be decomposed into the equivalent eigensystem with eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and corresponding unit eigenvectors $\hat{e}_1, \hat{e}_2, \hat{e}_3$. After rearranging,

$$\begin{aligned}\mathbf{S} =&(\lambda_1 - \lambda_2)(\hat{e}_1\hat{e}_1^T) + (\lambda_2 - \lambda_3)(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T) \\ &+\lambda_3(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T + \hat{e}_3\hat{e}_3^T).\end{aligned}$$

$\hat{e}_1\hat{e}_1^T$ is the *stick component*, $\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T$ is the *plate component*, and $(\hat{e}_1\hat{e}_1^T + \hat{e}_2\hat{e}_2^T + \hat{e}_3\hat{e}_3^T)$ is the *ball component* of $\mathbf{S}$. We define

- *surface saliency* by $\lambda_1 - \lambda_2$, with $\hat{e}_1$ indicating the normal direction,
- *curve saliency* by $\lambda_2 - \lambda_3$, with $\hat{e}_3$ indicating the tangent direction (normal to plate tensor), and
- *point saliency* by λ_3.

3.4 Maximal Surface Saliency Along a Line Segment ℓ

Therefore, after pass 1, the normal direction at a point P is available. This direction is given by $\hat{e}_1$ of the tensor vote accumulated at P after eigen-decomposition. To sample votes on a line segment ℓ (in pass 2 and 3), all Ps within the neighborhood of ℓ cast stick votes (by using 3D stick voting field as normals are now known) at quantized locations of ℓ. We project the surface saliency gradient onto its $\hat{e}_1$ vector, that is, by computing the inner product $q =< \hat{e}_1, \bigtriangledown(\lambda_1 - \lambda_2) >$, where $\bigtriangledown(\cdot)$ is the gradient vector on $\lambda_1 - \lambda_2$. *Maximal surface saliency* is detected by computing *zero crossing* along ℓ (Fig. 3a).

4 Algorithms for 3D Tensor Voting: Review

In this section, we give a review of the original second order tensor voting algorithm that appeared in [22]. Algorithms 1-4 are not difficult to implement. In the next section, we will describe the mathematics and tensor voting based algorithm for our adaptive estimation of tensors of curvature and conclude by Algorithm 5.

The voter uses GENTENSORVOTE (Algorithm 1) to cast a tensor vote to the vote receiver (votee). Stick votes generated by GENNORMALVOTE (Algorithm 2) are accumulated using COMBINE (Algorithm 3). A 3×3 $outTensor$ is the output. The votee thus receives a set of $outTensor$ from voters within its neighborhood. The resulting tensor matrices can be summed up by ADDTENSOR (Algorithm 4), which performs ordinary 3×3 matrix addition. The final matrix after accumulation describes a symmetric tensor in 3D. We make use of Algorithms 1-4, and Algorithm 5 later, to define the three passes of tensor voting.

Time complexity of each tensor voting pass is linear with the number of points. Detailed analyses are given in [14].

Using these algorithms, we state the three passes of our algorithm as follows (and illustrated in Fig. 1):

1. For each encoded input point (quantized at a grid center), receive ball votes generated by GENTENSORVOTE from all other (quantized) tokens within its neighborhood defined by σ in order to infer its normal direction. Because all input points are initially encoded as ball tensors (Section 3.1), GENTENSORVOTE is equal to the use of 3D ball voting field for casting ball votes (by setting `GenerateOtherComponents` to true).

   After vote interpretation (Section 3.3), obvious outliers (points with very low surface saliencies) and junctions (points with high curve or point junction saliencies) will be rejected. Outliers are points with very low surface saliencies and, thus, are likely noise. Junctions are indicated by high curve and point saliencies, where curvature estimation should not be performed.
2. To correct positions, we collect votes at grid corners and compute zero crossings along grid lines. This is achieved by computing the inner product q and labeling "+" or "-" at the grid corners, as illustrated in the 2D example shown in Fig. 3b. Zero crossings are located by linear interpolation along each grid line, which are connected together to obtain a local patch. The input point is projected along the normal direction onto this patch and the projected point on the patch is the location of the corrected token. Note that votes are collected at the grid corners of a

3×3 neighborhood (in 3D, it is $3 \times 3 \times 3$) since the gradient $\bigtriangledown(\lambda_1 - \lambda_2)$ should be computed. Curvature estimation is very sensitive to wrong locations of input points, especially quantized locations. Using the collected votes surrounding the quantized locations will better approximate the correct position at subvoxel precision. Here, the same GENTENSORVOTE is used to collect votes at the grid corners, except that `GenerateOtherComponents` is set to false as we now cast stick votes only to propagate the smoothness constraint as prescribed by the 3D stick voting field. The same vote interpretation is performed.

3. For each position-corrected inliers, we cast and collect *curvature tensor votes* using Algorithm 5 GENCURVVOTE, where a total of eight curvature tensor votes (at 45 degrees) are collected. The details are described in the next section.
4. (Optional) Finally, if a surface is to be obtained, curvature-based tensor voting fields can be used to vote for a surface at subvoxel precision to fill in missing information in the form of a surface [20], [21]. Misrejected outliers, if any, and gaps and misalignment produced by merging multiple laser scans are treated as missing data for filling. Recent work in [9] performs multiview surface matching.

Algorithm 1. GENTENSORVOTE (voter,votee)
GENNORMALVOTE is used to compute the most likely normal direction vote at the votee. Then, plate and ball tensors are computed by integrating the resulting normal votes cast by voter. In implementation, we only use GENTENSORVOTE *once* per given σ and store the resulting 3D stick and ball voting field into lookup tables.

for all $0 \leq i, j < 3$, outTensor[i][j] $\leftarrow$ 0
for all $0 \leq i < 2$,
  voterSaliency[i] $\leftarrow$ voter[λ_i] $-$ voter[λ_{i+1}]
voterSaliency[2] $\leftarrow$ voter[λ_2]
/* Generate the stick component */
if (voterSaliency[0] > 0) **then**
 vecVote $\leftarrow$ GENNORMALVOTE (voter,votee)
 COMBINE (outTensor,vecVote)
end if
transformVoter $\leftarrow$ voter
/* Generating other components of the tensor vote here */
if `GenerateOtherComponents` $=$ true **then**
 for $i = 1$ to 2 // 1: plate, 2: ball **do**
  if (voterSaliency[i] $>$ 0) **then**
   /* count[i] is a sufficient number of samples uniformly distributed on a unit sphere. */
   while (count[i] $\neq$ 0) **do**
    transformVoter[direction] $\leftarrow$ sample[direction] $\leftarrow$ GENUNIFORMDISTRIBUTEPT()
    if ($i \neq 2$) **then**
     /* Compute the alignment matrix, except the isotropic ball tensor */
     transformVoter[direction] $\leftarrow$ voter[eigenvectorMatrix] $\times$ sample[direction]
    end if
    vecVote $\leftarrow$ GENNORMALVOTE (transformVoter,votee)
    COMBINE (outTensor, vecVote, voterSaliency[i])
    count[i] $\leftarrow$ count[i] $-$ 1
   end while
  end if
 end for
end if
return outTensor

Algorithm 2. GENNORMALVOTE (voter, votee)
A vote (vector) on the most likely normal direction is returned.

$v \leftarrow$ votee[position] $-$ voter[position]
/* voter and votee are connected by high curvature? */
if (angle(voter[direction],v) $< \frac{\pi}{4}$) **then**
 return NULL {smoothness constraint violated}
end if
stickvote[position] $\leftarrow$ votee[position]
/* voter and votee on a straight line, or voter and votee are the same point */
if (angle(voter[direction],v) $= \frac{\pi}{2}$) **or** (voter $=$ votee) **then**
 stickvote[direction] $\leftarrow$ voter[direction]
 stickvote[length] $\leftarrow e^{\frac{-r^2}{\sigma^2}}$ {(3)}
 return stickvote
end if
Compute center and radius of the osculating hemisphere
/* assign stick vote */
stickvote[direction] $\leftarrow$ center $-$ votee[position]
stickvote[length] $\leftarrow e^{-(\frac{r^2+c\kappa^2}{\sigma^2})}$ {(3)}
return stickvote

Algorithm 3. COMBINE (tensorvote, stickvote, weight)
Tensor addition is performed, given a stick vote.

for all i, j such that $0 \leq i, j < 3$ **do**
 tensorvote[i][j] $\leftarrow$ tensorvote[i][j] + weight $\times$ stickvote[direction][i] $\times$ stickvote[direction][j]
end for

Algorithm 4. ADDTENSOR (outTensor, inTensor, weight)
Two second order symmetric tensors are added—simply matrix addition.

for all i, j such that $0 \leq i, j < 3$ **do**
 outTensor[i][j] $\leftarrow$ outTensor[i][j] + weight $\times$ inTensor[i][j]
end for

5 Curvature Tensor Estimation by Tensor Voting

Curvature tensor votes will be collected and summed up at the position-corrected tokens, by tensor voting in pass 3. Let us first describe how to decompose a curvature tensor into the corresponding principal curvatures and directions. Let P be a point on a regular surface S, k_1 and k_2 be the maximum and minimum curvatures, and $\hat{e}_1$ and $\hat{e}_2$ be the corresponding principal directions. Taubin [23] proposed to estimate a symmetric matrix at P that has eigenvectors $N_P, \hat{e}_1, \hat{e}_2$ and corresponding eigenvalues $0, k_1, k_2$. N_P is the normal at P. This matrix is defined by an integrand. The estimated linear map is in 2D, which lies on the tangent plane at P, denoted by $T_P(S)$. That is, a 2×2 symmetric matrix M_p is defined that has eigenvectors $\hat{e}_1, \hat{e}_2$ with respect to the tangent space at $T_P(S)$ and eigenvalues k_1, k_2.

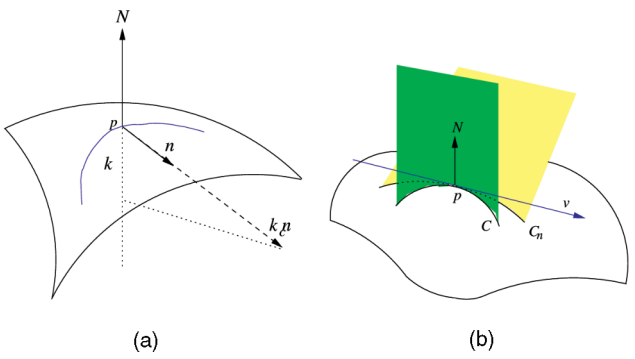

Fig. 4. (a) Normal curvature and (b) Meusnier theorem: The normal curvatures obtained from C and C_n are the same.

Our curvature tensor estimation directly estimates M_p. In the orthonormal basis $\{\hat{e}_1, \hat{e}_2\}$, we have [23]:

$$M_P = \frac{1}{2\pi}\int_{-\pi}^{\pi} k_n(t) t t^T dt, \quad t \in T_P(S) \tag{4}$$

$$= E^T \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} E, \quad E = [\hat{e}_1\ \hat{e}_2]^T, \tag{5}$$

where t is a tangent at P lying on $T_P(S)$ and $k_n(t)$ is the *normal curvature* at P given by the second fundamental form. After some algebraic manipulation, we can obtain the values of m_{11}, m_{12}, m_{21}, and m_{22}: $m_{11} = \frac{3}{8}k_1 + \frac{1}{8}k_2, m_{12} = 0, m_{22} = \frac{1}{8}k_1 + \frac{3}{8}k_2$, and $m_{21} = 0$. Therefore, upon decomposing the curvature tensor into its eigensystem, the principal curvatures are given by $k_1 = 3m_{11} - m_{22}$ and $k_2 = 3m_{22} - m_{11}$ and the corresponding principal directions are given by $\hat{e}_1$ and $\hat{e}_2$. Note that a related formulation could be found in [11]. In the following, we describe Algorithm 5 or the third process in Fig. 1: estimation of the tensor of curvatures by tensor voting.

5.1 Definition of Curvature Tensor Vote

First, let us begin by stating the result of this section by relating the tensor voting approach with Taubin [23] and Page et al. [15]. The notations here will be referred back in our description later. By (2), a curvature tensor vote is given by $w_{ij}k_{ij}t_{ij}t_{ij}^T$:

$$v_i = r \text{ (votee : vote receiver)} \tag{6}$$

$$v_j = c \text{ (voter : vote caster)} \tag{7}$$

$$w_{ij} = 1 \tag{8}$$

$$k_{ij} = sign(c)k_{rc} \tag{9}$$

$$t_{ij} = t. \tag{10}$$

The variables on the right-hand side of the above equations will be explained. After v_i has accumulated (by tensor sum) all the curvature tensor votes cast by v_j, a curvature tensor is obtained. By decomposing it into the corresponding eigensystem having eigenvalues $m_{11} \geq m_{22}$ and eigenvectors $\hat{e}_1, \hat{e}_2$, we can obtain the principal curvatures and directions at v_i.

In the following, we first relate and explain the underlying differential geometry theory on normal curvatures [3] upon which the design of our curvature tensor votes is grounded.

5.2 Normal Curvatures

First, refer to Fig. 4. Let C be a regular curve in S passing through $P \in S$, k_c be the curvature of C at P, and $\cos\theta =$

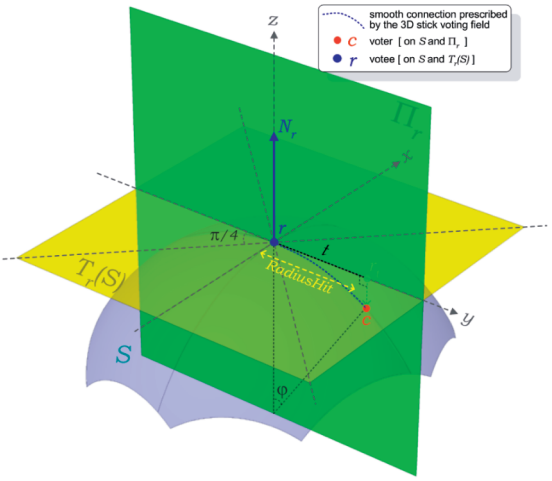

Fig. 5. Principal direction and curvature votes.

$< n, N >$ be the inner product of n and N, where n is the unit normal vector to C and N is the unit normal vector of S at P. The value $k = k_c \cos\theta$ is called the *normal curvature* [3], Fig. 4a. Therefore, k is the length of the projection of the vector $k_c n$ onto the line defined by N. Meusnier theorem [3] states that *all* curves lying on a surface S and having the same tangent line at P have the *same* normal curvature, Fig. 4b. Refer to Fig. 5. In tensor voting, the estimation of principal curvatures and directions is translated into:

- (tensor) the casting of a curvature tensor vote tt^T, with magnitude k_{rc}, from a voter c to a vote receiver r, along a tangent line t at r and
- (voting) the accumulation or summation of these votes to estimate the curvature tensor at r. The accumulated tensor will be decomposed into principal curvatures and directions as described at the beginning of this section.

Given a tangent line t, the Meusnier theorem [3] implies that we can choose *any* planar curve C such that t is tangential to C at r and C and N are lying on the same plane. Therefore, the curvature of C at r is representative and is equal to the normal curvature at r.

Since an input mesh is unavailable, we postulate the value of curvature by hypothesizing a smooth connection between r and c using the 3D stick voting field. Note that the normal at r is known after pass 1 so that we can align the stick field to vote for smooth connection. We pick one orientation and let the normal vector be N_r. In the following, we explain our definition of curvature tensor votes, which consists of curvature sign, $sign$, magnitude value, k_{rc}, and direction tensor, tt^T. They correspond to k_{ij} and $t_{ij}t_{ij}^T$, respectively, in (2). Note that t or t_{ij} can be treated as 2-vectors lying on the tangent plane if we define the local 3D coordinate system at r with the z-axis parallel the normal at r, thus making the z-component zero. To normalize the vote contribution along a direction, we equalize each contribution cast by a voter, that is, $w_{ij} = 1$, and collect curvature tensor vote at 45 degrees apart. All curvature tensor votes are defined and collected on the tangent plane, which is the same as the x-y plane of the local frame. As will be shown in the next section, the curvature magnitude is related to the position of c, which is voted for by tensor voting by sampling normal tensor votes along the z-direction at r_t (defined shortly by (11)).

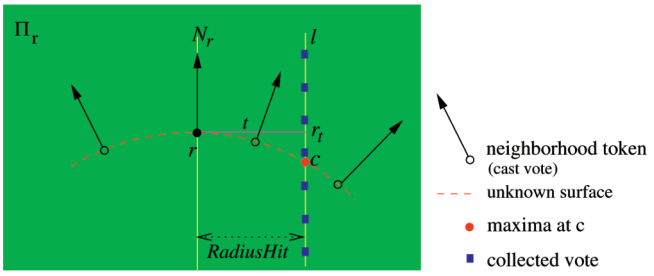

Fig. 6. $RadiusHit$ of a curvature tensor: vote for maximum surface saliency without a mesh. $b = ||r_t - c||$ is defined after ℓ has been fixed by N_r (given in pass 1) and $RadiusHit$ (parameter).

5.3 $RadiusHit$ of an Adaptive Curvature Tensor

We propose using the $RadiusHit$ of a curvature tensor to adjust the neighborhood size and adapt to local feature scale for enhancing estimation accuracy. In this section, we first explain the $RadiusHit$ of a curvature tensor. Estimation accuracy is explained in the next section.

$RadiusHit$ defines the range where quantized *normal tensor votes* (different from *curvature tensor votes*), given by GENTENSORVOTE, should be sampled. Also, $RadiusHit$ implicitly encodes the local feature scale, making our curvature estimation adaptive to different feature scales. Thus, our method automatically preserves fine or high curvature details and, hence, increases estimation accuracy.

By the definition of normal curvatures and Meusnier theorem, we choose a planar curve that is incident at r, having tangent t, and lies on the plane Π_r (Π_r is the plane perpendicular to and intersect $T_r(S)$ along t), Fig. 5. Recall that in this configuration, the normal curvature is equal to the curvature of the planar curve.

Refer to Fig. 6. Select a point r_t along the tangent line t at r by fixing a scalar parameter called $RadiusHit$:

$$r_t = r + t \cdot RadiusHit. \tag{11}$$

Given $RadiusHit$, we define a line segment ℓ (Fig. 6):

$$\ell = r_t + \rho \cdot N_r, -RadiusHit \leq \rho \leq RadiusHit. \tag{12}$$

In the above equations, note that r_t, r, t, ℓ, and N_r are all 3-vectors.[1] Instead of explicit surface fitting or extraction, we estimate the subvoxel location c along ℓ, where maximal surface saliency s occurs by zero-crossing detection, that is, $\frac{ds}{d\ell} = 0$ at c.

The estimation of the maxima c is performed by tensor voting. Refer to Fig. 6 again for illustration. Tokens associated with normal directions (available after pass 1) in the vicinity of ℓ are aligned with the *3D stick voting field*, in order to cast *normal tensor votes* (GENTENSORVOTE) at quantized locations along ℓ. Each quantized location obtains its *surface saliency* s. *Zero crossing* in $\frac{ds}{d\ell}$ is then detected at subvoxel precision. This less expensive zero crossing detection is similar to the zero crossing detection in [14], except that no surface is generated for efficiency. All italic terms in this paragraph are already explained in Section 3.

After estimating the subvoxel location c where the maxima c occurs, the curvature of the smooth connection between r and c is given by $sign(c)k_{rc}$, where $sign(c) = 1$ if c is above $T_r(S)$ and $sign(c) = -1$ if c is below $T_r(S)$. If c lies on $T_r(S)$, $sign(c) = 0$. k_{rc} is

$$k_{rc} = \left| \frac{2\cos\frac{\pi-\varphi}{2}}{||r-c||} \right|, \tag{13}$$

which is derived from the reciprocal of the radius of curvature, where φ, r, and c are defined as in Fig. 5.

Define $b = ||r_t - c||$. After some algebraic manipulation, (13) becomes

$$k_{rc}(b) = \frac{2b}{b^2 + RadiusHit^2}. \tag{14}$$

Therefore, given a $RadiusHit$, the accuracy of $k_{rc}(b)$ is completely characterized by the accuracy of b. $b = ||r_t - c||$ is thus our independent variable for evaluating the accuracy and applicability of our method. Note that c is estimated independently by tensor voting, while r_t is known after fixing $RadiusHit$.

5.4 Summary

This section is summarized by Algorithm 5 GENCURVVOTE, which produces the estimated principal curvatures and directions. The construction has been illustrated with the help of Fig. 5 and Fig. 6.

Algorithm 5. GENCURVVOTE (r, N_r, $RadiusHit$)
INPUT r is the corrected position, N_r is the normal at r, $RadiusHit$
OUTPUT k_1, k_2 (maximum, minimum principal curvatures) p_1, p_2 (maximum, minimum principal curvature directions)

$votedir \leftarrow$ any direction at $r \perp N_r$
$\mathbf{M}_r \leftarrow$ zero 2×2 matrix
for $i = 0$ to 7 **do**
  $votedir \leftarrow$ rotate($votedir$, $\frac{\pi}{4}$) on tangent plane $T_r(S)$
  $r_t \leftarrow r + votedir \cdot RadiusHit$
  $c \leftarrow$ detectMaxima($r_t + N_r \cdot RadiusHit, r_t - N_r \cdot RadiusHit$) {Use GENTENSORVOTE to sample (vote for) surface saliency along ℓ in Fig. 6, given the $RadiusHit$.}
  if $c =$ no maxima **then**
    return NULL {No curvature}
  end if
  $b \leftarrow$ length($r_t - c$)
  $k_{rc} \leftarrow \frac{2b}{b^2 + RadiusHit^2}$ {Compute directional curvature}
  $\mathbf{M}_r \leftarrow \mathbf{M}_r + \frac{1}{8} \cdot k_{rc} \cdot (votedir \cdot votedir^T)$
end for
$(\lambda_1 , \lambda_2 , \hat{e}_1 , \hat{e}_2) \leftarrow$ eigenDecompose($\mathbf{M}_r$)
$k_1 \leftarrow 3\lambda_1 - \lambda_2$, $k_2 \leftarrow 3\lambda_2 - \lambda_1$
$p_1 \leftarrow \hat{e}_1$, $p_2 \leftarrow \hat{e}_2$
return (k_1 , k_2 , p_1 , p_2)

6 Estimation Accuracy and Applicability

Now, we discuss the effect of choice of $RadiusHit$ on the accuracy of the estimated curvatures $k_{rc}(b)$. We use a cylinder for experimentation, which is an interesting shape in curvature estimation as many approaches overestimate (respectively, underestimate) k_2 (respectively, k_1). Fig. 7 plots the relationship between b and $k_{rc}(b)$ for $RadiusHit = 1, 2, 3, 4$.

Since curvature is a differential property, it is valid to assume the constancy of curvature along the geodesic path from r to c. We have the following observations:

1. Recall that they are expressed in the 3D coordinate system at r with the z-axis parallel to N_r and where the x-axis and y-axis are any orthonormal vectors lying on the tangent plane $T_r(S)$.

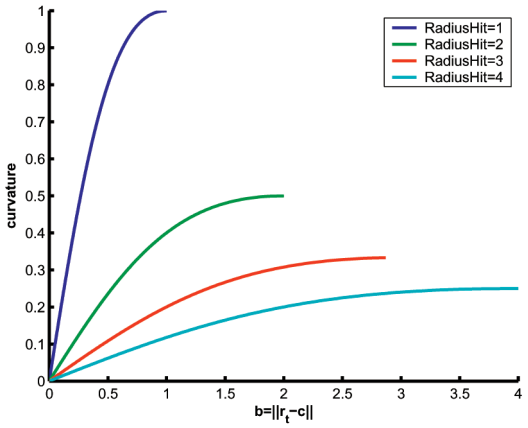

Fig. 7. Estimation accuracy and applicability of a curvature tensor: plot curvature versus $b = ||r_t - c||$ (6) for $RadiusHit = 1, 2, 3, 4$, curvature $= k_{rc}(b) = \frac{2b}{b^2 + RadiusHit^2}$.

1. **Accuracy**. Recall that b is the only variable affected by tensor voting, so it best quantifies the accuracy of our tensor voting algorithm, where quantization or outlier noise may lead to an inexact b. From (14), we therefore define the curvature error as

$$err(k_{rc}) = k_{rc}(\hat{b}) - k_{rc}(b), \tag{15}$$

where, $\hat{b} = b + \delta b$. For a given error δb, *the smaller the* $RadiusHit$, *the larger the* $err(k_{rc})$ *is*, see Fig. 7. This is manifested by the larger gradient for small $RadiusHit$ at a given b. Note that, in Fig. 7, as the $RadiusHit$ increases, the sensitivity of curvature change decreases given the same difference in b, thus reducing the fluctuation of errors in estimation when there exists estimation errors in b.

2. **Applicability**. However, the range of curvatures k_{rc} that can be estimated decreases with increasing $RadiusHit$. This observation is in fact related to local feature scale. Observe that the largest curvature that can be estimated with $RadiusHit = 4$ is only 0.25, while it is increased to 1 for $RadiusHit = 1$.

In fact, the maximum curvature estimable by a given $RadiusHit$ is simply its reciprocal (i.e., $\max(k_{rc}(b)) = \frac{1}{RadiusHit}$). Refer to Fig. 8a, the maximum applicable $RadiusHit$ is equal to the radius of osculating circle of the underlying curve. It is clear that at $RadiusHit = 5$, a maxima can no longer be "hit." By definition, since the 3D stick voting field hypothesizes a smooth connection from r to c by using an osculating circle (Section 3), the values b can take is also constrained by the $RadiusHit$, thus making $\max(b) = RadiusHit$.

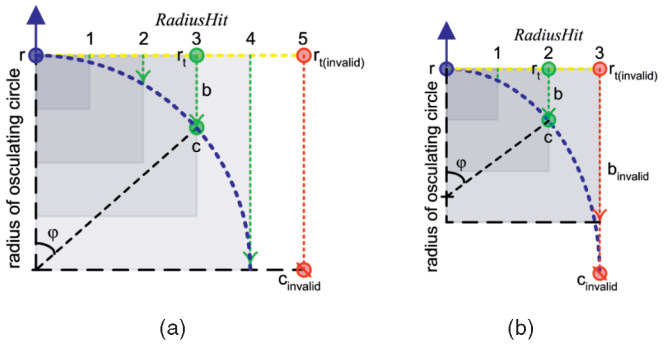

Fig. 8. Applicability of $RadiusHit$. The gray area defines the applicability range of a given $RadiusHit$.

In Fig. 8b, when $RadiusHit = 3$, it is clear that $b_{invalid} = ||r_{t(invalid)} - c_{invalid}|| > RadiusHit$, invalidating $RadiusHit = 3$. Thus, it is sufficient to search for the maxima c along ℓ, whose signed distance from r_t is in $[-RadiusHit, RadiusHit]$, that is, (12). The quantization step size along ℓ is made proportional to $RadiusHit$ to allow for a higher accuracy. Typical values $RadiusHit$ can take are 2 and 3 and the quantization step size for ρ is usually set to be the reciprocal of $RadiusHit$.

Ideally, if noise is not problematic, $RadiusHit$ at r should be as small as possible ($= 1$ or within resolution limit) to capture the differential property of the changing curvature. However, when noises are present, $RadiusHit$ should be increased to reduce the noise effect by gathering more normal tensor votes in the supporting neighborhood and to capture curvature details at proper scale.

Observation (1) below gives a concrete algorithm on choosing the locally optimal $RadiusHit$ (if exists) for accurate estimation. See Table 2 for meanings of notations.

1. *Fixed noise, varying* $RadiusHit$. Refer to Fig. 9a. For the synthetic data TORUS, ground truth is available for k_1^t, which is equal to -0.3333 for every point. So, let us use k_1^e for illustration. In the frequency plot, the narrower the spread centered at -0.3333, the better the curvature estimation is. Observe that the spread is narrow when the appropriate $RadiusHit$ ($= 2$) is chosen. When the $RadiusHit$ is too large or too small, the accuracy of k_1^e is affected. Observe further that a contiguous range of $RadiusHit$ gives good k_1^e.

The above observation indicates how to choose $RadiusHit$: For a given input token, run our tensor

TABLE 2
Some Notations Used in the Paper

Notation	Meaning	Notation	Meaning
k_1^e (k_1^t)	estimated (true) maximum curvature	n^e (n^t)	estimated (true) normal direction
p_1^e (p_1^t)	estimated (true) maximum direction	x^e (x^t)	estimated (true) x-coordinate
k_2^e (k_2^t)	estimated (true) minimum curvature	y^e (y^t)	estimated (true) y-coordinate
p_2^e (p_2^t)	estimated (true) minimum direction	z^e (z^t)	estimated (true) z-coordinate

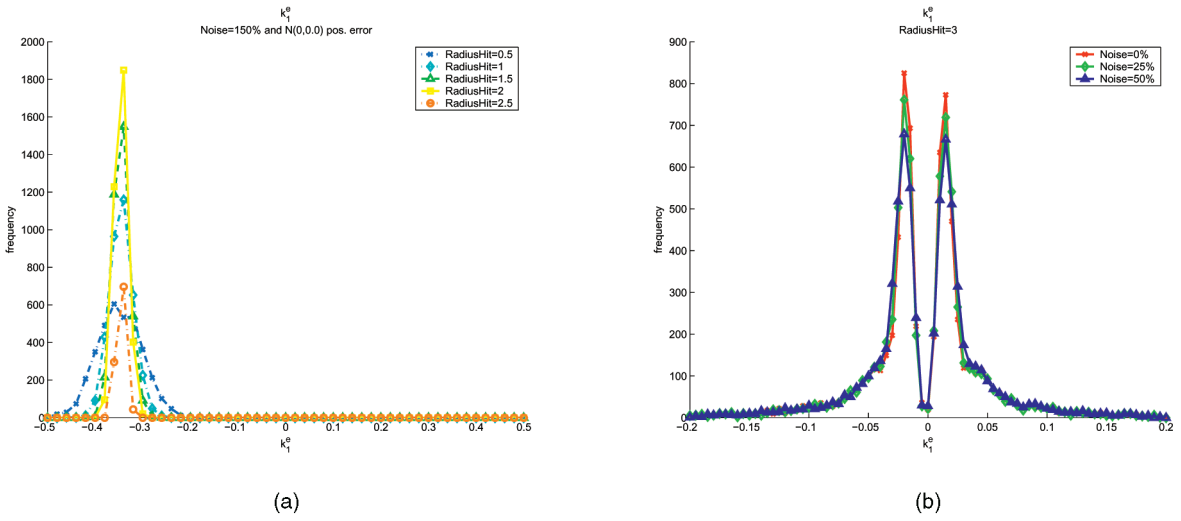

Fig. 9. (a) TORUS with fixed amount of outlier noise (150 percent) with varying $RadiusHit$. (b) MANNEQUIN HEAD with fixed $RadiusHit$ and varying amount of noise. Only k_1^e frequency distributions are shown. Similar plots were obtained for k_2^e. See Table 2 for meanings of notations.

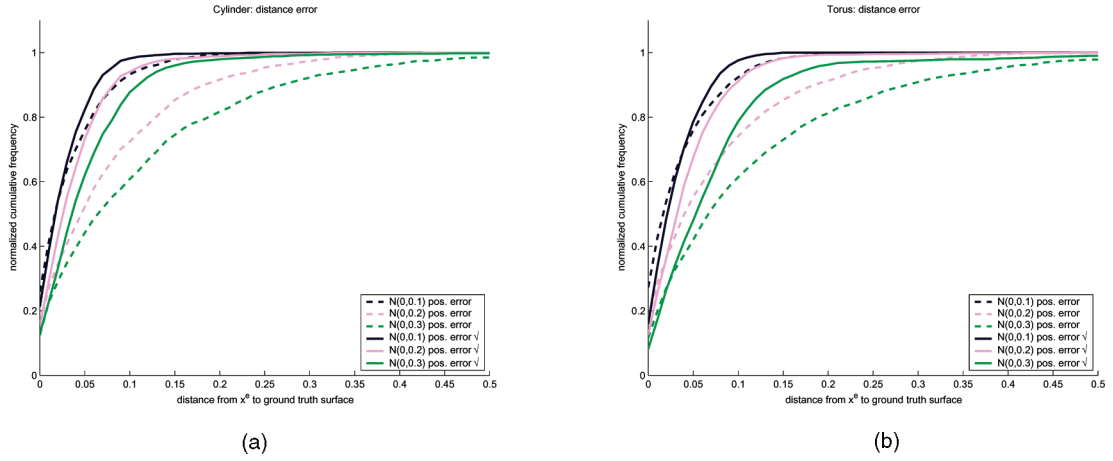

Fig. 10. Evaluation of the robustness of our tensor voting method in the presence of different level of misalignment/quantization *and* outlier noise together (pass 2): distance from ground truth surface error analysis, for (a) CYLINDER and (b) TORUS. The (optimal) $RadiusHit$ chosen are ~ 3 and ~ 2, respectively. Dashed curves show the position error before correction, solid curves show error the after the correction. The normalized cumulative frequency is computed by accumulating the frequency from 0 percent error to 100 percent error in steps of 5 percent, normalized by the total number of estimated results.

voting curvature estimation algorithm on several choices of $RadiusHit$ at a given input site. Pick the $RadiusHit$ with the largest peak in the frequency plot. Similar findings are obtained for other data we used, where several peaks still exist for different k_1^t (and, thus, k_1^e) in the presence of noise. For real data, although no ground truth is available, we still observe several curves that represent a range of $RadiusHit$ still cluster together, producing a salient peak in the frequency plot for k_1^e. Similar discussion applies to k_2^e.

Based on the above observation, we formulate the problem of choosing the optimal $RadiusHit$, $RadiusHit_{opt}$, as one of minimizing the variance of the curvature sample distribution:

$$RadiusHit_{opt} = argmin_{RadiusHit} variance(RadiusHit), \tag{16}$$

where $variance(\cdot)$ is the variance of the curvature sample distribution for a given $RadiusHit$. Since the number of estimations (or equivalently the number of inliers and corrected outliers) is the same for all $RadiusHit$, it follows statistically that the $RadiusHit$ with the minimum variance is the most reliable parameter for curvature estimation.

2. *Varying noise, fixed RadiusHit*. Fig. 9b (real data MANNEQUIN HEAD) shows that our approach is very insensitive to the amount of outliers. Observe that the shape of the frequency distribution of the curvatures remain relatively invariant to an increasing amount of outlier noise.

7 Quantitative and Qualitative Evaluation

We run detailed experiments to show the robustness and accuracy of our curvature tensor estimation method by tensor voting. Quantization noise (or more generally, misalignment noise[2]) and outlier noise are directly addressed here, which are problematic to many curvature estimation approaches, especially second-order derivative

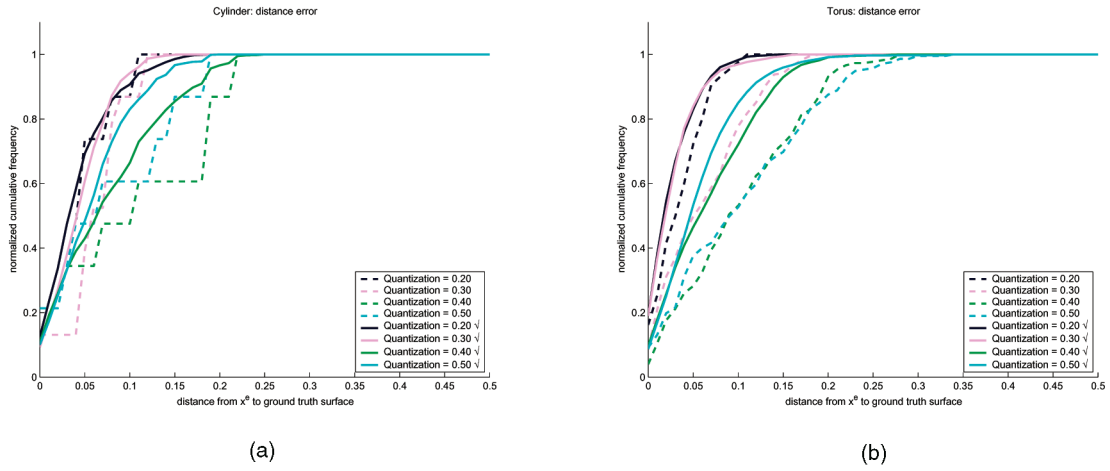

Fig. 11. Evaluation of the robustness of our tensor voting method in the presence of different level of explicit grid quantization *and* outlier noise together (pass 2): distance from ground truth surface analysis, for (a) CYLINDER and (b) TORUS. The (optimal) $RadiusHit$ chosen are ~ 3 and ~ 2, respectively. Dashed curves show the position error before correction, solid curves shows the error after the correction.

methods. Synthetic and real data are used. The notations used in the plots are explained in Table 2. All the figures in this section are color coded.

7.1 Misalignment Error and Outlier Noise

Since we vote for maximum surface saliency by detecting local maxima in subvoxel precision, our curvature tensor estimation overcomes the inherent difficulty associated with data quantization/misalignment.

In pass 2, we eliminate misalignment/quantization error by shifting the noisy points to the underlying surface, which is nonetheless *not* explicitly extracted because a mesh input is unnecessary. Curvature tensor votes are estimated at the position-corrected points in pass 3. Although reconstruction of mesh using Voronoi methods for general 3D data clouds and simple 2D grid adjacency for $2\frac{1}{2}$D range finder data are fast, they are sensitive to error, while high quality mesh reconstruction using level sets may be time consuming to construct.

We first experimented with synthetic data CYLINDER and TORUS because ground truths are available ([15], [23] also used them) and, therefore, quantitative analysis and comparison can be made. Qualitative analysis for real data will be followed. In all cases, we used the optimal $RadiusHit$ whenever it exists (that is, salient peaks in frequency plots exist), computed using the procedure described in Section 6.

The robustness and accuracy of pass 1 has been evaluated elsewhere [14]. Let us evaluate the corrective ability of tensor voting in the presence of misalignment noise (that is, pass 2). We evaluate the robustness of tensor voting in the presence of *both* misalignment/quantization and outlier noise. In the following examples, 150 percent salt and pepper noise is added into the bounding boxes of the testing cases. That is, the signal to noise ratio is 100 to 150, or equivalent for every 100 true data points, there are 150 noisy data points. Further, misalignment/quantization noise is added. In Fig. 10, the misalignment noise we used follows normal distribution $N(0, \sigma)$ with zero mean and variance σ, where σ is 0.1, 0.2, and 0.3, respectively. In Fig. 11, the misalignment noise we used follows an explicit grid-based quantization with grid sizes 0.1 to 0.5. After the first two passes of tensor voting, we are able to eliminate most of the outlier noise and correct the erroneous points. Any perturbation to true locations larger than the above are treated as outliers. In our method, tensor voting will correct the position of outliers as much as possible, or else reject them if the errors are too large.

We can conclude from Fig. 10 and Fig. 11 the very good corrective ability of our method: In the plots, the solid curves (for position corrected tokens) indicate much less distance to

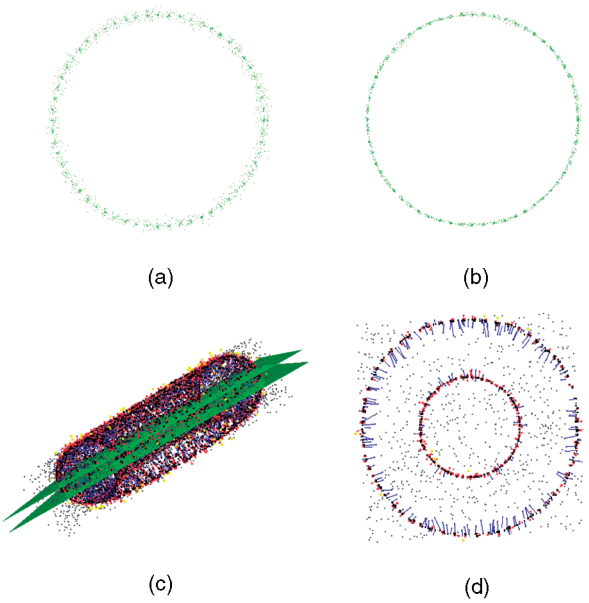

Fig. 12. Qualitative evaluation of the correction ability by pass 2 tensor voting for *synthetic* data. See text for color codes. The planes in (c) show the cross sections. (a) Top view of CYLINDER: perturbed tokens. (b) Position-corrected tokens. (c) Zoom out view of TORUS: perturbed tokens and outlier noises. (d) Top view of TORUS: cross-sectional view of position-corrected tokens.

2. As will be shown in our experiments, the input point locations are offset from the ground truth location by random or normal distributions to produce misalignment noise. Quantization noise can be regarded as a special case of such misalignment errors when the perturbed locations are at (integral) grid locations.

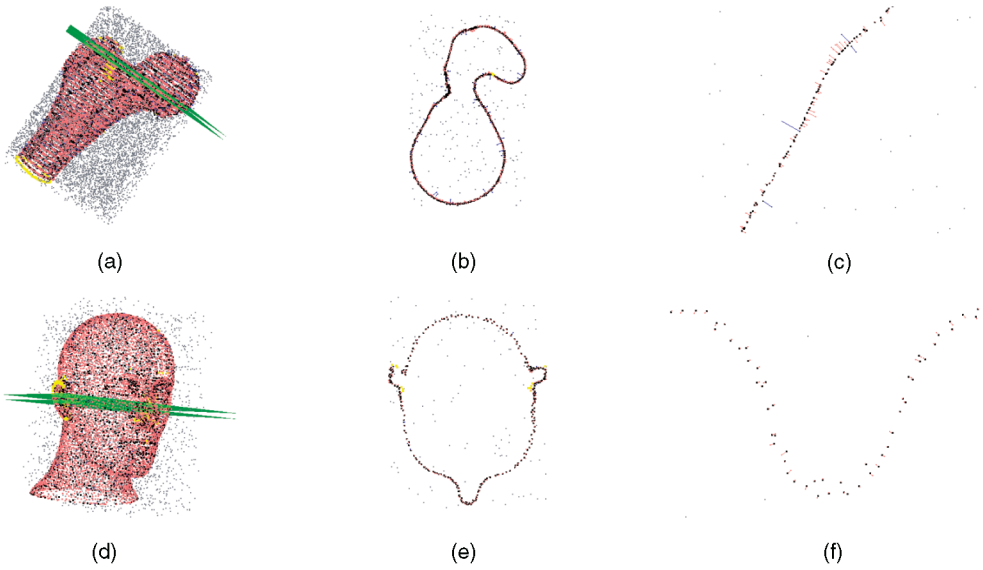

Fig. 13. Qualitative evaluation of the token position correction ability by pass 2 tensor voting for *real* data. First column, noisy data with the location of cross sections shown (zoom out views). Second column, Cross sectional views (see text for color codes). Third column, Cross sectional views (zoom in views, see text for color codes).

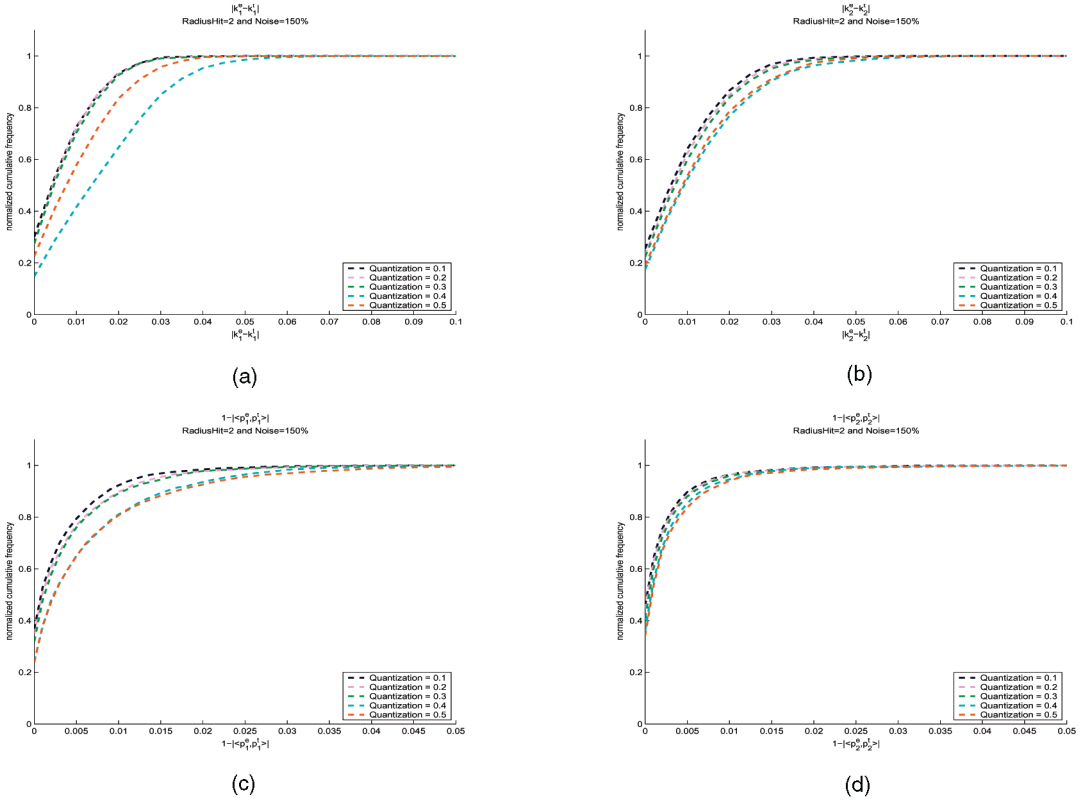

Fig. 14. TORUS. Result on varying different quantization grid sizes and fixed outlier noise level of $150\ percent$ with $RadiusHit \approx 2$. (a) $|k_1^e - k_1^t|$, (b) $|k_2^e - k_2^t|$, (c) $1 - | < p_1^e, p_1^t > |$, and (d) $1 - | < p_2^e, p_2^t > |$.

ground truth surface error than that of the dashed curves (tokens with perturbed positions). A similar situation applies in all solid-dashed curve pairs in the same color.

We now qualitatively evaluate the robustness using synthetic (Fig. 12) and real data (Fig. 13). White points are corrected locations where the curvature tensors are computed. Pink points are supposed to be ground truth points, but they are misaligned or perturbed by Gaussian noise $N(0, 0.3)$, where 150 percent of salt and pepper noises are added as well. Light gray points are noisy points that cannot be associated with any curvature estimation and, thus, they are identified as uncorrectable outliers. Blue points are noisy points, but they can barely adopt the curvature value in their vicinities. Yellow points are points that cannot be corrected and, thus, classified as outliers where curvature tensor votes are not collected.

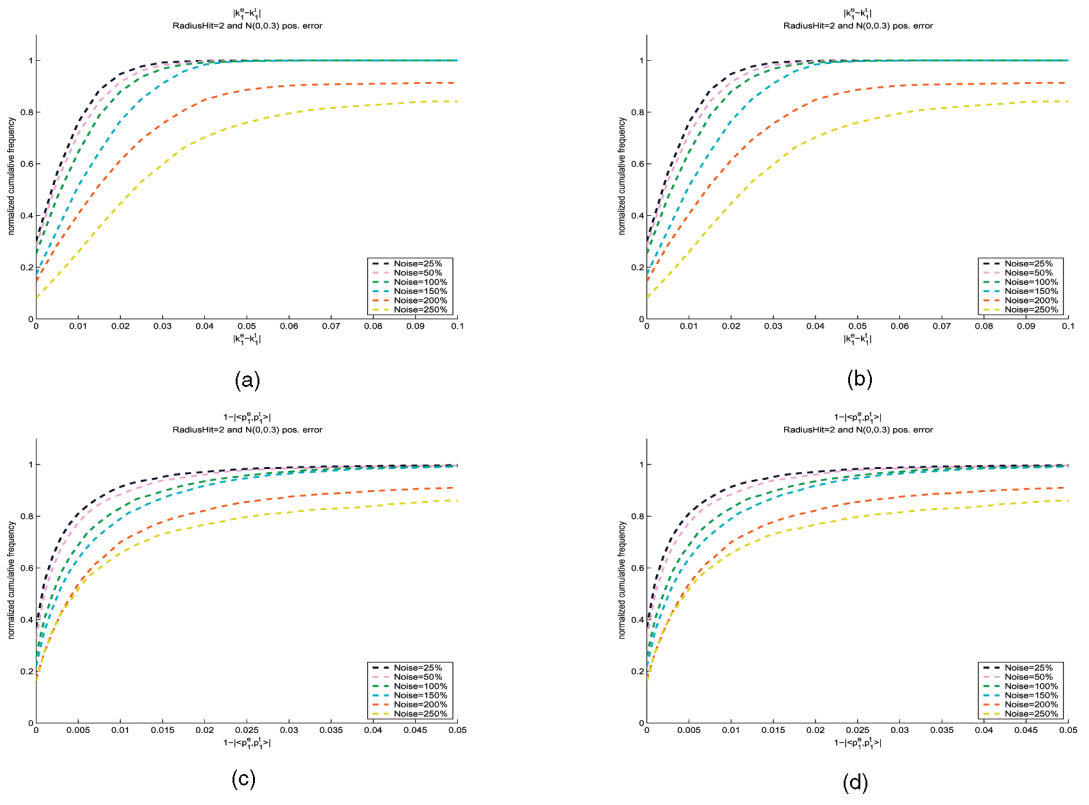

Fig. 15. TORUS. Quantitative result on varying outlier noise and fixed Gaussian noise $N(0, 0.3)$ with $RadiusHit \approx 2$. (a) $|k_1^e - k_1^t|$, (b) $|k_2^e - k_2^t|$, (c) $1 - | < p_1^e, p_1^t > |$, and (d) $1 - | < p_2^e, p_2^t > |$.

7.2 Accuracy of Curvature Tensor Estimation

We have evaluated our algorithm on synthetic data using a cylinder and a torus. Only the results for torus are shown here as it is a more challenging case when noise is present. Note that curvature tensors will *not* be estimated at identified uncorrectable outliers. Moreover, in pass 3, curvature tensors will be estimated at the position-corrected tokens (the inliers) to avoid misalignment errors. In the following plots, we illustrate the curvature tensor accuracy by exhausting the following combinations:

1. Varying grid quantization level with optimal $RadiusHit$ and fixed outlier noise level of 150 percent (Fig. 14).
2. Varying outlier noise level with optimal $RadiusHit$ and fixed Gaussian noise level of $N(0, 0.3)$ (Fig. 15).
3. Varying $RadiusHit$ with fixed outlier noise level of 150 percent and Gaussian noise level of $N(0, 0.3)$ (Fig. 16).

More analysis on the following combinations can also be found in [25]:

1. On the three representative rings of TORUS with varying outlier noise level of 50 percent, 150 percent, and with quantization grid size of 0.3.
2. Varying Gaussian noise, with optimal $RadiusHit$ and fixed outlier noise level of 150 percent.

Since ground truths are available for synthetic data, the analysis shown in the plots are performed by pairwise comparison between estimated and ground truth values. In the cases of varying Gaussian noise and outlier noise level (case (1) and (2)), our algorithm is very robust and degrades very gracefully. In most experiments, we have the highest frequencies for small error, i.e., fast saturation in the normalized cumulative frequency plot. Considering the large amount of noise we have also added, even the largest errors are, in fact, rather low, which also have the lowest frequencies. Also, compared with Taubin [23] on the same torus example, the frequencies of small error (within the bucket size of the histograms) in our experiments are highest, while in [23] no noise was considered.

As for variation in $RadiusHit$ (case (3)), the results show that curvature estimation accuracy increases with increasing $RadiusHit$, but up to a certain limit. In particular, for TORUS, the accuracy drops when $RadiusHit = 2.5$ since ℓ (Fig. 6) fails to hit the torus surface when $RadiusHit$ is too large. A large number of inliers are also filtered out. Thus, all results confirm with (1) and (2) in Section 5.3 on the choice of $RadiusHit$.

Figs. 17, 18, and 19 show the qualitative results of FEMUR, TOYOTA, and MOKONA. Two other results, THEO and MANNEQUIN, can be found in [25]. For the FEMUR real data set, the noise level is all 50 percent (that is, there exists one noisy point for every two data points). The $RadiusHit$ we used is 2. For each data set, the noisy input is also shown. The curvature directions, signs of the estimated principal curvatures, mean curvatures, or Gaussian curvatures are plotted (together with the underlying surface, which is for contextual visualization only). TOYOTA was acquired by a low quality laser scanner and MOKONA was acquired by low quality laser pointer scanner. In both real data sets, no additional noisy data points were added.

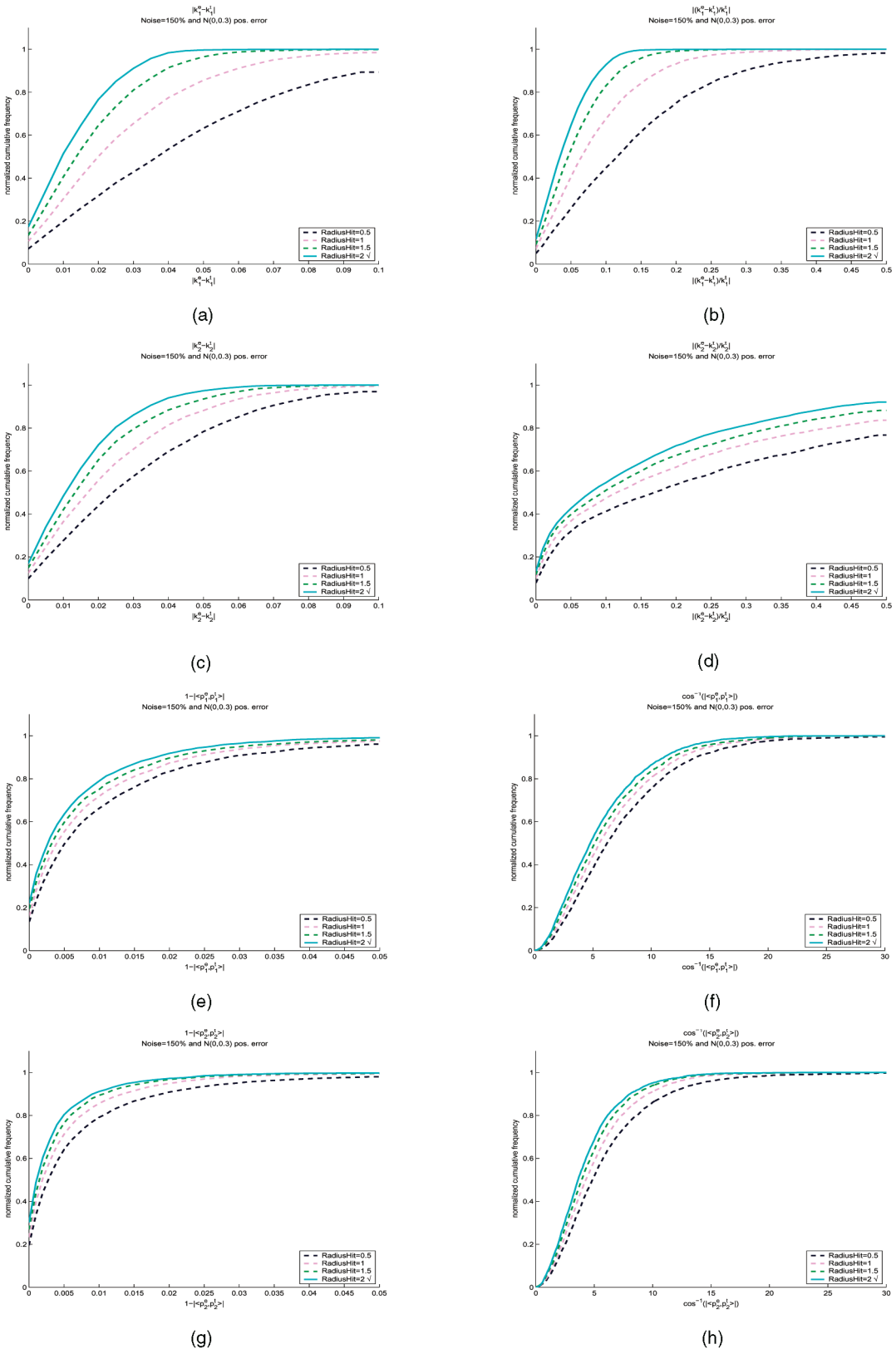

Fig. 16. TORUS. Quantitative result on varying $RadiusHit$ with fixed Gaussian noise $N(0, 0.3)$ and outlier noise level of 150 percent. (a) $|k_1^e - k_1^t|$, (b) $|(k_1^e - k_1^t)/k_1^t|$, (c) $|k_2^e - k_2^t|$, (d) $|(k_2^e - k_2^t)/k_2^t|$ (if $k_2^t = 0, \text{just}|(k_2^e - k_2^t)|$), (e) $1 - | < p_1^e, p_1^t > |$, (f) $\cos^{-1}(| < p_1^e, p_1^t > |)$, (g) $1 - | < p_2^e, p_2^t > |$, and (h) $\cos^{-1}(| < p_2^e, p_2^t > |)$.

It is expected that if a mesh is available, our results are comparable to that of Taubin [23] and Page et al. [15] on the same input mesh. However, when the input consists of an unorganized set of points in the presence of noise, our method does not need a mesh to compute curvature information.

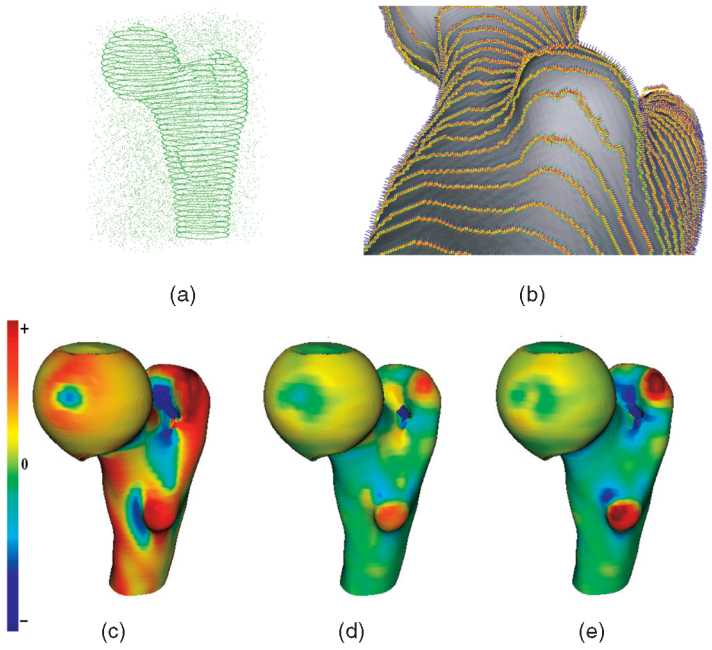

Fig. 17. FEMUR. The tokens are shown with the underlying surface for contextual visualization. The curvature are color-coded as the scale on the left, with gradual changes from high positive curvature (red) to zero curvature (green) to high negative curvature (blue).

8 CONCLUSION

We described a three-pass tensor voting algorithm to perform quantitative curvature tensor estimation adapting to local feature scales from which principal curvatures and directions are calculated. In the previous two-pass algorithm [20], [21], quantization is not addressed and thus only qualitative estimation (signs of Gaussian curvature) can be reliably obtained. In this paper, we robustly vote for the curvature tensors at position-corrected tokens. In essence, our method first uses tensor voting for normal estimation. The main technical contribution consists of vote sampling at quantized locations to correct point location (pass 2), thus avoiding

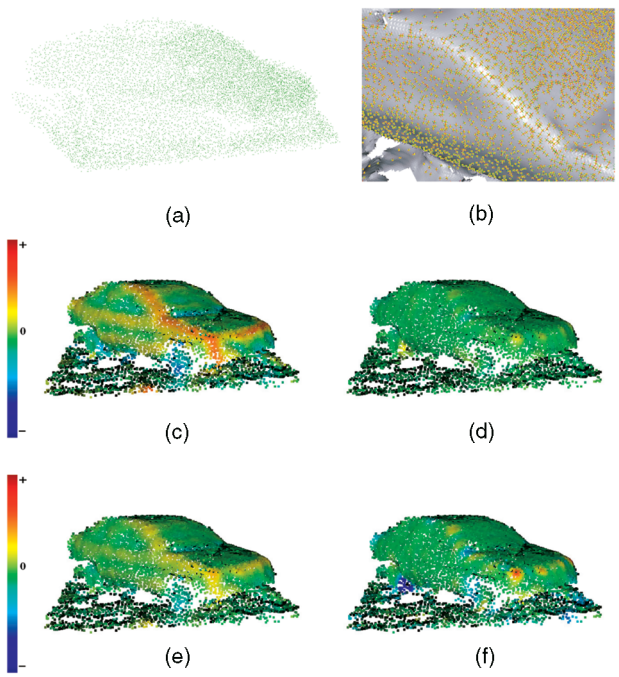

Fig. 18. TOYOTA. In (b) the tokens are shown with the underlying surface for contextual visualization. The curvature are color-coded as the scale on the left, with gradual changes from high positive curvature (red) to zero curvature (green) to high negative curvature (blue).

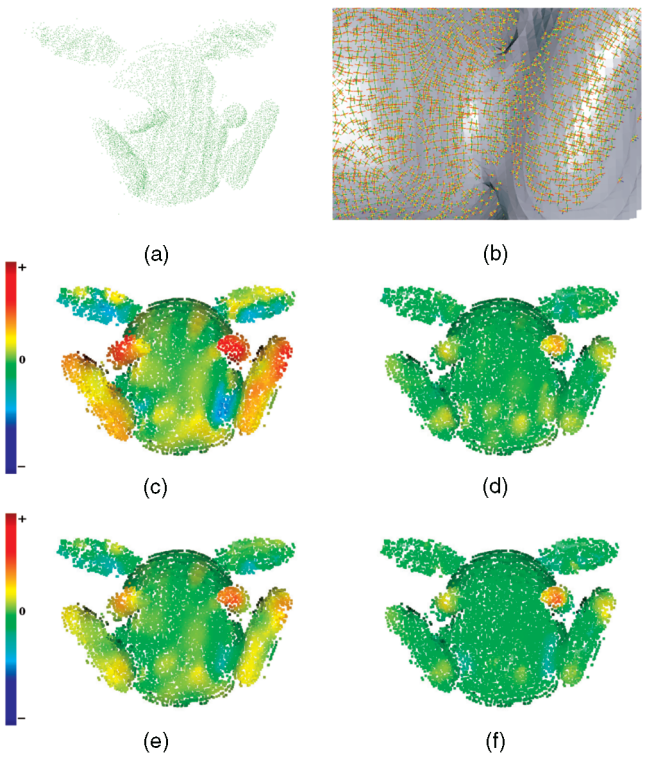

Fig. 19. MOKONA. In (b) the tokens are shown with the underlying surface for contextual visualization. The curvature are color-coded as the scale on the left, with gradual changes from high positive curvature (red) to zero curvature (green) to high negative curvature (blue).

expensive surface fitting and the definition of the tensor voting algorithm for curvature tensor estimation (pass 3), which is in agreement with differential geometry theories. Further, we qualitatively and quantitatively evaluate our method from the following perspectives: corrective ability (to overcome quantization errors), curvature estimation robustness (in the presence of quantization and/or outlier noise), and different feature scales (as demonstrated in real and synthetic data). We compared our approach with other curvature tensor estimation methods and performed detailed qualitative and quantitative experiments on noisy and complex inputs.

In the future, we hope to advance on the theoretical level, possibly by establishing the relationship between tensor voting and Bayesian or maximum likelihood methods.

Acknowledgments

The authors would like to thank all the reviewers for their constructive comments to improve the final manuscript. This research is supported by the Research Grant Council of Hong Kong Special Administrative Region, China: HKUST6193/02E.

References

[1] E. Angelopoulou and L.B. Wolff, "Photometric Computation of the Sign of Gaussian Curvature Using a Curve-Orientation Invariant," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. 432-437, 1997.

[2] P.J. Besl and R.C. Jain, "Segmentation through Variable-Order Surface Fitting," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 2, pp. 167-192, Mar. 1988.

[3] M.P.D. Carmo, *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.

[4] I. Douros and B.F. Buxton, "Three-Dimensional Surface Curvature Estimation Using Quadric Surface Patches," *Scanning 2002 Proc.,* May 2002.

[5] G. Dudek and J. Tsotsos, "Shape Representation and Recognition from Multi-Scale Curvature," *Computer Vision and Image Understanding,* pp. 170-189, 1997.

[6] T.J. Fan, *Describing and Recognizing 3-D Objects Using Surface Properties.* Springer-Verlag, 1989.

[7] P.J. Flynn and A.K. Jain, "On Reliable Curvature Estimation," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR '89)* pp. 110-116, 1989.

[8] M. Charlebois, K. Gupta, and S. Payandeh, "Curvature Based Shape Estimation Using Tactile Sensing," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 3502-3507, 1996.

[9] D. Huber and M. Hebert, "Fully Automatic Registration of Multiple 3D Data Sets," *Image and Vision Computing,* vol. 21, no. 7, pp. 637-650, July 2003.

[10] S. Karbacher and G. Hausler, "A New Approach for Modeling and Smoothing of Scattered 3D Data in Three-Dimensional Image Capture and Applications," *SPIE Proc.,* pp. 168-177, 1998.

[11] N. Khaneja, M.I. Miller, and U. Grenander, "Dynamic Programming Generation of Curves on Brain Surfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1260-1265, 1998.

[12] R. Malladi and I. Ravve, "Fast Difference Schemes for Edge Enhancing Beltrami Flow," *Proc. Seventh European Conf. Computer Vision,* pp. I:343-357, 2002.

[13] R. Martin, "Estimation of Principal Curvatures from Range Data," *Int'l J. of Shape Modeling,* pp. 99-109, 1998.

[14] G. Medioni, M.-S. Lee, and C.-K. Tang, *A Computational Framework for Segmentation and Grouping.* Elsevier Science Inc, 2000.

[15] D.L. Page, A. Koschan, Y. Sun, J. Paik, and M.A. Abidi, "Robust Crease Detection and Curvature Estimation of Piecewise Smooth Surfaces from Triangle Mesh Approximations Using Normal Voting," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition,* pp. I:162-167, 2001.

[16] S. Petitjean, "A Survey of Methods for Recovering Quadrics in Triangle Meshes," *ACM Computing Surveys,* vol. 2, no. 34, pp. 211-262, 2002.

[17] B. Rieger, F.J. Timmermans, L.J. van Vliet, and P.W. Verbeek, "Curvature Estimation of Surfaces in 3-D Grey-Value Images," *Proc. 16th Int'l Conf. Pattern Recognition,* 2002.

[18] P.T. Sander and S.W. Zucker, "Stable Surface Estimation," *Proc. Int'l Conf. Pattern Recognition,* pp. 1165-1167, 1986.

[19] P.T. Sander and S.W. Zucker, "Inferring Surface Trace and Differential Structure from 3-D Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 9, pp. 833-854, Sept. 1990.

[20] C.K. Tang and G. Medioni, "Robust Estimation of Curvature Information from Noisy 3D Data for Shape Description," *Proc. Seventh IEEE Int'l Conf. Computer Vision,* pp. 426-433, 1999.

[21] C.K. Tang and G. Medioni, "Curvature-Augmented Tensor Voting for Shape Inference from Noisy 3D Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 6, pp. 858-864, June 2002.

[22] C.K. Tang, G. Medioni, and M.S. Lee, "N-Dimensional Tensor Voting and Application to Epipolar Geometry Estimation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 8, pp. 829-844, Aug. 2001.

[23] G. Taubin, "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation," *Proc. Fifth IEEE Int'l Conf. Computer Vision,* pp. 902-907, 1995.

[24] W.S. Tong and C.K. Tang, "Multiscale Tensor Voting," technical report, The Hong Kong University of Science and Technology, http://www.cs.ust.hk/~cstws/research/tv_multiscale, 2004.

[25] W.S. Tong and C.K. Tang, "Robust Estimation of Adaptive Tensors of Curvature by Tensor Voting Results," technical report, The Hong Kong University of Science and Technology, http://www.cs.ust.hk/~cstws/research/tv_curvature, 2004.

[26] E. Trucco and R.B. Fisher, "Experiments in Curvature-Based Segmentation of Range Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 2, pp. 177-182, Feb. 1995.

[27] B.C. Vemuri, A. Mitiche, and J.K. Aggarwal, "Curvature-Based Representation of Objects from Range Data," *Image and Vision Computing,* vol. 4, pp. 107-114, May 1986.

[28] M. Yang and E. Lee, "Segmentation of Measured Point Data Using a Parametric Quadric Surface Approximation," *Computer-Aided Design,* pp. 449-458, 1999.

[29] P. Yuen, N. Khalili, and F. Mokhtarian, "Curvature Estimation on Smoothed 3-D Meshes," *Proc. British Machine Vision Conf.,* pp. 133-142, Dec. 1999.

Wai-Shun Tong received the BEng, MPhil, and PhD degrees from the Hong Kong University of Science of Technology in computer science. He was a visiting student at the University of Southern California (USC), Los Angeles, in 2001. In 2002, he was awarded the Microsoft Fellowship Asia. In 2003, he was a visiting student at the Visual Computing Group of Microsoft Research Asia. He is currently a post-doctoral fellow in the Vision and Graphics Group at the Hong Kong University of Science and Technology. His research interests include multiscale analysis, surface reconstruction, stereo and motion analysis, shape analysis, data visualization, and vision and graphics topics. He is a student member of the IEEE Computer Society.

Chi-Keung Tang received the MS and PhD degrees in computer science from the University of Southern California (USC), Los Angeles, in 1999 and 2000, respectively. He has been with the Computer Science Department at the Hong Kong University of Science and Technology since 2000, where he is currently an assistant professor. He is also an adjunct researcher at the Visual Computing Group of Microsoft Research, Asia, working on various exciting research topics in computer vision and graphics. His research interests include low to midlevel vision, such as segmentation, correspondence, shape analysis, and vision and graphics topics, such as image-based and video-based rendering. He is a member of the IEEE Computer Society.