Tensor Voting: Theory and Applications

Gérard Medioni
Institute for Robotics & Intelligent Systems
USC, Los Angeles, CA 90089
medioni@iris.usc.edu

Chi-Keung Tang
Computer Science Department
HKUST, Hong Kong
chitang@iris.usc.edu

Mi-Suen Lee
Philips Research
Briarcliff Manor, NY 10510
misuen.lee@philabs.philips.com

Abstract

We present a unified computational framework which properly implements the smoothness constraint to generate descriptions in terms of surfaces, regions, curves, and labelled junctions, from sparse, noisy, binary data in 2-D or 3-D. Each input site can be a point, a point with an associated tangent direction, a point with an associated normal direction, or any combination of the above. The methodology is grounded on two elements: tensor calculus for representation, and linear voting for communication: each input site communicates its information (a tensor) to its neighborhood through a predefined (tensor) field, and therefore casts a (tensor) vote. Each site collects all the votes cast at its location and encodes them into a new tensor. A local, parallel marching process then simultaneously detects features. The proposed approach is very different from traditional variational approaches, as it is non-iterative. Furthermore, the only free parameter is the size of the neighborhood, related to the scale. We have developed several algorithms based on the proposed methodology to address a number of early vision problems, including perceptual grouping in 2-D and 3-D, shape from stereo, and motion grouping and segmentation, and the results are very encouraging.

1 Introduction

In computer vision, we often face the problem of identifying salient and structured information in a noisy data set. From greyscale images, edges are extracted by first detecting subtle local changes in intensity and then linking locations based on the noisy signal responses. From binocular images, surfaces are inferred by first obtaining depth hypotheses for points and/or edges using local correlation measures, and then selecting and interpolating appropriate values for all points in the images. Similarly, in image sequence analysis, the estimation of motion and shape starts with local measurements of feature correspondences which give noisy data for the subsequent computation of scene information. Hence, for a salient structure estimator to be useful in computer vision, it has to be able to handle the presence of multiple structures, and the interaction between them, in noisy, irregularly clustered data sets. In this paper, we present a novel and unified methodology for the robust inference of features from noisy data. The organization of this paper is as follows: Section 2 gives an overview of previous work. Section 3 describes our overall tensor voting formalism. In Section 4 we present examples on feature inference from noisy 2-D and 3-D data. Then, in Section 5, we extend the basic formalism to solve early vision problems, such as shape from stereo, and motion grouping and segmentation, and also visualization problems such as flow visualization. Finally, we conclude in Section 6.

2 Previous Work

Based on the type of input, we can broadly classify previous work on the inference of curves, regions, and surfaces into 3 areas, namely, dot clustering in 2-D, curve and contour inference in 2-D, and surface reconstruction in 3-D.

In 2-D, Zucker and Hummel [33] address the problem of region inference from dot clusters using iterative relaxation labeling technique. Recently, Shi and Malik [24] propose to treat region inference as a graph partitioning problem, and present a recursive algorithm that uses the normalized cut as the global criterion to segment tokens into regions. Later, Parent and Zucker [20] use a relaxation labeling scheme that imposes co-circularity and constancy of curvature. Dolan and Weiss [4] demonstrate a hierarchical approach to grouping relying on compatibility measures such as proximity and good continuation. Sha'ashua and Ullman [23] propose the use of a saliency measure to guide the grouping process, and to eliminate erroneous features in the image. The scheme prefers long curves with low total curvature by using an incremental optimization scheme (similar to dynamic programming).

In 3-D, the problem is to fit sufaces to a cloud of possibly noisy points. Successful approaches include the physics-based approaches [29], in which the initial pattern deforms iteratively to the desired shape, subject to an objective function. Poggio and Girosi's [21] network learning approach can be used when we know in advance the pattern consists of a single surface. Fua and Sander [6] propose a local algorithm to describe surfaces from a set of points. Szeliski *et al.* [25] propose the use of a physically-based dynamic local process evolving from each data point (particle), and subject to various forces. They are able to handle objects of unrestricted topology, but assume a single connected object. Hoppe *et al.* [13] and Boissonnat [3] use computational geometry to address the problem by treating the data as vertices of a graph and constructing edges based on local properties. Another approach, alpha-shapes [5], has also attracted much attention in the computational geometry community. Recently, a new approach, known as the level-set approach [32], has produced very good results, and attracted significant attention. It allows changes in topology. Other methodologies employ the perceptual saliency approach, such as the works by Thornber and Williams [31], and by Sarkar and Boyer [22], Sha'ashua and Ullman [23]. Among all the methods, Guy and Medioni's work is one of the non-iterative approaches that is able to handle both curve inference in 2-D [9] and surface inference in 3-D from point, curve element, or surface patch element inputs [10], although individually. Unlike other methods mentioned here, their method is robust to a considerable amount of noise, has no limitation on the surface topology, detects orientation discontinuity, and is non-iterative.

3 The tensor voting formalism

An overall illustration of our method, summarizing its different components, is shown in Figure 1, which shows the 3-D version. The methodology is grounded on two elements: *tensor calculus* for data representation, and linear tensor *voting* for data communication. Each input site propagates its information in a neighborhood. The information is encoded in a tensor, and is determined by a predefined voting field. Each site collects the information cast there and analyzes it, building a saliency map for each feature type. Salient features are located at local extrema of these saliency maps, which can be extracted by non-maximal suppression.

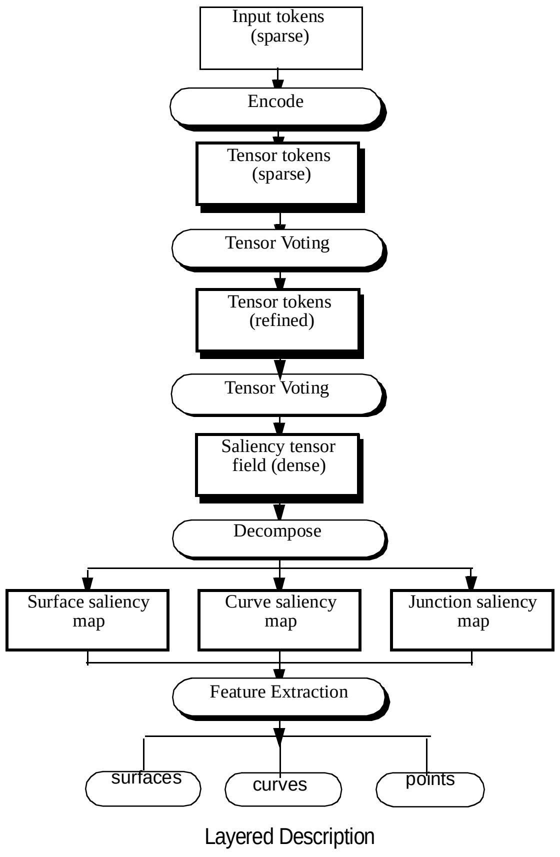


Figure 1. Overall approach

Each input token is first encoded into a second order symmetric tensor. For instance, if the input token has only position information, it is transformed into an isotropic tensor (a ball) of unit radius.

In a first voting stage, tokens communicate their information with each other in a neighborhood, and refine the information they carry. After this process, each token is now a generic second order symmetric tensor, which encodes confidence of this knowledge (given by the tensor size), curve and surface orientation information (given by the tensor orientations).

In a second stage, these generic tensor tokens propagate their information in their neighborhood, leading to a dense tensor map which encodes feature saliency at every point in the domain. In practice, the domain space is digitized into a uniform array of cells. In each cell the tensor can be decomposed into elementary components which express different aspects of the information captured.

The resulting dense tensor map is then decomposed. Surface, curve, and junction features are then obtained by extracting, with subvoxel precision, local extrema of the corresponding saliency values along a direction. The final output is the aggregate of the outputs for each of the components.

3.1 Representation

Our goal is to extract geometric structures such as regions, curves, surfaces, and the intersection between them. These are well defined mathematical entities with the well known properties.

Points can simply be represented by their coordinates. A first order local description of a curve is given by the point coordinates, and its associated tangent or normal. A second order description would also include the associated curvature.

A first order local description of a surface patch is given by the point coordinates, and its associated normal. A second order description would also include the principal curvatures and their directions.

Here, however, we do not know in advance what type of entity (point, curve, surface) a token may belong to. Furthermore, because features may overlap, a location may actually correspond to both a point and a curve, or even to a surface, a curve and a point *at the same time*. An example of such a case occurs at the intersection between 2 smooth curves: the intersection should be a point, that is, it represents a junction with no associated tangent information, but also represents 2 curve elements, as the curves do not stop there.

3.2 Second order symmetric tensor

To capture first order differential geometry information and its singularities, a *second order symmetric tensor* is used. It captures both the orientation information and its confidence, or *saliency*. Such a tensor can be visualized as an ellipse in 2-D, or an ellipsoid in 3-D. Intuitively, the shape of the tensor defines the type of information captured (point, curve, or surface element), and the associated size represents the saliency. For instance, in 2-D, a very salient curve element is represented by a thin ellipse, whose major axis represents the estimated tangent direction, and whose length reflects the saliency of the estimation.

To express a second order symmetric tensor ***S***, graphically depicted by an ellipse in 2-D, and an ellipsoid in 3-D, we choose to take the associated quadratic form, and to diagonalize it, leading to a representation based on the eigenvalues λ_1, λ_2,λ_3 and the eigenvectors $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$, $\hat{\boldsymbol{e}}_3$. In a more compact form, $\boldsymbol{S} = \lambda_1\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\lambda_2\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T+\lambda_3\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^T$, where $\lambda_1\geq\lambda_2\geq\lambda_3\geq 0$ are the eigenvalues, and $\hat{\boldsymbol{e}}_1$, $\hat{\boldsymbol{e}}_2$, $\hat{\boldsymbol{e}}_3$ are the eigenvectors corresponding to λ_1, λ_2, λ_3 respectively. Note that, because *S* is a second order symmetric tensor, the eigenvalues are real and positive (or zero), and the eigenvectors form an orthonormal basis.

The eigenvectors correspond to the principal directions of the ellipsoid/ellipse and the eigenvalues encode the size and shape of the ellipsoid/ellipse, as shown in Figure 2. ***S*** is a *linear* combination of outer product tensors and, therefore a tensor.

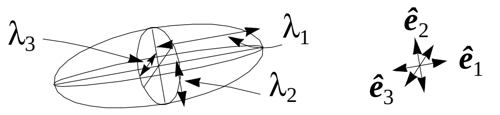


Figure 2. An ellipsoid and its eigensystem

3.3 Tensor decomposition

All the tensors used so far are somewhat singular, in the sense that the associated ellipsoid has the shape of either a stick, a plate, or a ball. As a result of the voting procedure (to be described later), we produce *arbitrary* second-order, symmetric tensors, therefore we need to handle any generic tensor.

The spectrum theorem [8] states that any tensor can be expressed as a *linear* combination of these 3 cases, i.e., $\boldsymbol{S}=(\lambda_1-\lambda_2)\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+(\lambda_2-\lambda_3)(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T)+\lambda_3(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T+\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^T)$, where $\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T$ describes a stick, $(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T)$ describes a plate, and $(\hat{\boldsymbol{e}}_1\hat{\boldsymbol{e}}_1^T+\hat{\boldsymbol{e}}_2\hat{\boldsymbol{e}}_2^T+\hat{\boldsymbol{e}}_3\hat{\boldsymbol{e}}_3^T)$ describes a ball. Figure 3 shows the decomposition of a general saliency tensor into the stick, plate, and ball components.

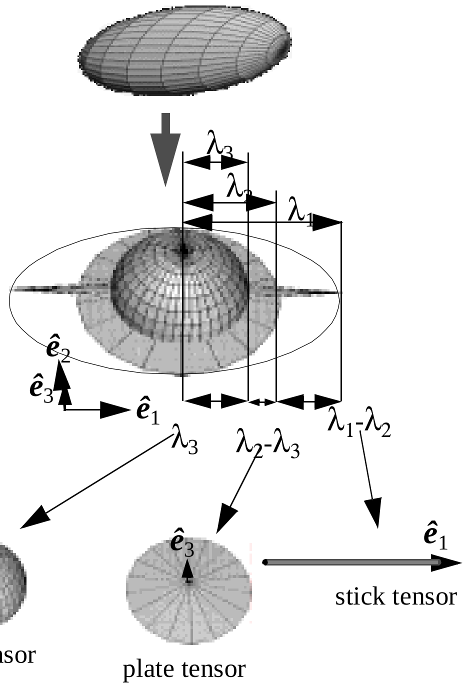


Figure 3. tensor decomposition

At each location, the estimate of each of the 3 types of information, and their associated saliency, is therefore captured as follows:

point-ness: no orientation, saliency is λ_3

curve-ness: orientation is $\hat{\boldsymbol{e}}_3$, saliency is λ_2 -λ_3

surface-ness: orientation is $\hat{\boldsymbol{e}}_1$, saliency is λ_1-λ_2

In 2-D, there is no surface-ness, and curve-ness is expressed by $\hat{\boldsymbol{e}}_1$ for the tangent orientation, and by λ_1-λ_2 for curve saliency.

We have now explained the information encoded in a second order symmetric tensor, which consists of three independent elements, and a measure of feature saliency.

3.4 Tensor communication

We now turn to our communication and computation scheme, which allows a site to exchange information with its neighbors, and infer new information.

Token refinement and dense extrapolation. The input tokens are first encoded as perfect tensors. In 3-D, a point token is encoded as a 3-D ball. A point associated with tangent direction is encoded as a 3-D plate. A point associated with normal direction is encoded as 3-D stick. These initial tensors communicate with each other in order to

- derive the most preferred orientation information, or refine the initial orientation if given, for each of the input tokens (*token refinement),* and

- extrapolate the above inferred information at every location in the domain for the purpose of coherent feature extraction (*dense extrapolation*).

In the token refinement case, each token collects all the tensor values cast at its location by all the other tokens. The resulting tensor value is the tensor sum of all the tensor votes cast at the token location.

In the dense extrapolation case, each token is first decomposed into its independent elements, and broadcasts this information, using an appropriate tensor field, which also defines a neighborhood, and puts the corresponding tensor value at every location. In practice, values are entered at discrete cell locations. The tensor value at any given location in the domain is the *tensor sum* of all the tensor votes cast at this location. A somewhat subtle difference occurs in this second case, as ball tensors define isolated features, which therefore do not need to propagate their information, and thus do not vote.

While they may be implemented differently for efficiency, these 2 operations are equivalent to a *voting* process, and can be regarded as *tensor convolution* with *voting kernels*, and produces tensors in turn. This tensor convolution is in fact a simple alignment process, involving a translation followed by a rotation.

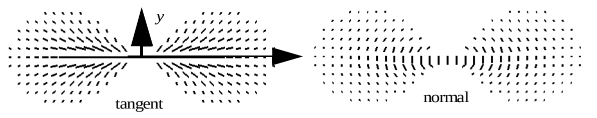
Therefore, in 3-D, it remains to describe the derivation of the 3-D voting kernels.

All voting kernels can be derived from the *fundamental 2-D stick kernel*, by rotation and integration. Figure 4 shows this 2-D stick kernel. In [18], we explain in mathematical terms that this voting kernel in fact encodes the proximity and the smoothness constraints. Note that a direction can be defined by either the tangent vector, or the normal vector, which are orthogonal to each other. We can therefore define two equivalent fundamental fields, depending whether we assign a tangent or normal vector at the receiving site.

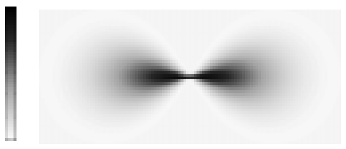
Derivation of the 3-D voting kernels. Denote the fundamental 2-D stick kernel by V_F. In 3-D, we need 3 voting kernels: the stick, the plate, and the ball voting kernels. The 3-D stick kernel is obtained by revolving the normal version of V_F 90 degrees about the *z*-axis (denote it by V_F'). Then, we rotate V_F' about the *x*-axis, and integrate the contributions by tensor addition during the rotation. The resulting 3-D stick kernel describes the orientation $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ in world coordinates. The other orientations can be obtained by a simple rotation. Hence, w.l.o.g., the 3-D stick kernel is defined mathematically as

$$\int_0^{\pi} V'_F d\alpha \Big|_{\beta = 0,\, \gamma = 0} \qquad (1)$$

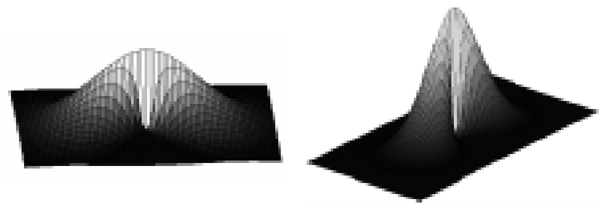
where α, β, γ are angles of rotation about the *x*, *y*, *z* axis respectively.



orientation



intensity-coded strength (saliency)



2 views of the field strength of 2-D stick field

Figure 4. The fundamental 2-D stick field

To obtain the plate kernel, we rotate the 3-D stick kernel obtained above about the *z*-axis, integrating the contributions by tensor addition. Therefore, the 3-D plate kernel thus obtained describes a plate with normal orientation is $\begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$. Again, the other orientations can be obtained readily by a simple rotation. Denote the 3-D stick kernel by *V*, the plate is derived from *V* by:

$$\int_0^{\pi} V d\gamma \Big|_{\alpha = 0,\, \beta = 0} \qquad (2)$$

To obtain the ball kernel, we rotate the 3-D stick kernel about the *y*-axis and *z*-axis (the order does not matter), integrating the contributions by tensor addition:

$$\int_0^{\pi} V d\beta d\gamma \Big|_{\alpha = 0} \qquad (3)$$

3.5 Feature Extraction

In this section, we summarize the marching process for feature extraction in 3-D, by giving mathematical definitions. Detailed implementation can be found in [18].

At the end of the voting process, we have produced a dense tensor map, which is then decomposed in two *dense vector maps* (three in 3-D).

In 3-D, each voxel of these maps has a 2-tuple $(s, \hat{\boldsymbol{v}})$, where s is a scalar indicating strength and $\hat{\boldsymbol{v}}$ is a unit vector indicating direction:

- *Surface map* (*SMap*): $s = \lambda_1 - \lambda_2$, and $\hat{\boldsymbol{v}} = \hat{\boldsymbol{e}}_1$ indicates the normal direction.
- *Curve map* (*CMap*): $s = \lambda_2 - \lambda_3$, and $\hat{\boldsymbol{v}} = \hat{\boldsymbol{e}}_3$ indicates the tangent direction.
- *Junction map* (*JMap*): $s = \lambda_3$, and $\hat{\boldsymbol{v}}$ is arbitrary.

These maps are dense vector fields, which are then used as input to our extremal algorithms in order to generate features such as junctions, curves, and surfaces.

The definition of point extremality, corresponding to junctions, is straightforward: it is a local maximum of the scalar value s.

Surface extremality. Let each voxel in the given vector field hold a 2-tuple $(s, \hat{\boldsymbol{n}})$ where s indicates field strength and $\hat{\boldsymbol{n}}$ denotes the normal direction. The vector field is continuous and dense, i.e., $(s, \hat{\boldsymbol{n}})$ is defined for every point $\boldsymbol{p}$ in 3-D space.

A point is on an *extremal surface* if its strength s is locally extremal along the direction of the normal, i.e., $ds/d\hat{\boldsymbol{n}} = 0$. This is a necessary condition for 3-D surface extremality. A sufficient condition, which is used in implementation, is defined in terms of zero crossings along the line defined by $\hat{\boldsymbol{n}}$ .We therefore define the gradient vector $\boldsymbol{g}$ as, $\boldsymbol{g} = \left[\partial s/\partial x, \partial s/\partial y, \partial s/\partial z\right]^T$ and project $\boldsymbol{g}$ onto $\hat{\boldsymbol{n}}$, i.e., $q = \hat{\boldsymbol{n}} \cdot \boldsymbol{g}$. Thus, an extremal surface is the locus of points with $q = 0$.

Curve extremality. Each voxel in the given potential vector field holds a 2-tuple $(s, \hat{\boldsymbol{t}})$, where s is the field strength and $\hat{\boldsymbol{t}}$ indicates the tangent direction. Suppose the field is continuous and dense, in which $(s, \hat{\boldsymbol{t}})$ is defined for every point $\boldsymbol{p}$ in 3-D space.

A point $\boldsymbol{p}$ with $(s, \hat{\boldsymbol{t}})$ is on an *extremal curve* if any displacement from $\boldsymbol{p}$ on the plane normal to $\hat{\boldsymbol{t}}$ will result in a lower s value, i.e.,where u and v define the plane $\partial s/\partial u = \partial s/\partial v = 0$ normal to $\hat{\boldsymbol{t}}$ at $\boldsymbol{p}$ .This is a necessary condition for 3-D curve extremality. A sufficient condition, which is used in implementation, is defined in terms of zero crossings in the *u-v* plane normal to $\hat{\boldsymbol{t}}$. Define $\boldsymbol{q} = \mathbf{R}(\hat{\boldsymbol{t}} \times \boldsymbol{g})$ where $\mathbf{R}$ defines a rotation to align a frame with the *u-v* plane and $\hat{\boldsymbol{g}}$ is defined earlier. By construction, an extremal curve is the locus of points with $\boldsymbol{q} = \mathbf{0}$.

4 Feature inference in 2-D and 3-D

In this section, we apply the tensor voting formalism for structure inference from 2-D and 3-D data.

4.1 2-D

Here, we consider the general case where the input consists of both oriented and non-oriented data. Both types of data can be handled exactly the same way using our unified methodology, by having a first pass in which the oriented tokens vote with the 2-D fundamental stick kernel, and the non-oriented tokens with a 2-D ball kernel. Votes are collected only at the locations where tokens appear. Then, a second pass of voting is performed. Here, we discard the ball component of the tensor, and dense votes are collected. A dense tensor map is produced. This dense tensor map is then decomposed into a junction map and a curve map. Figure 5 shows an example of curve and junction inference from a mixed input consisting of an ellipse made up of curve elements (tangents), and a curve made up of dots. Another example of a "pretzel" is shown in Figure 6.

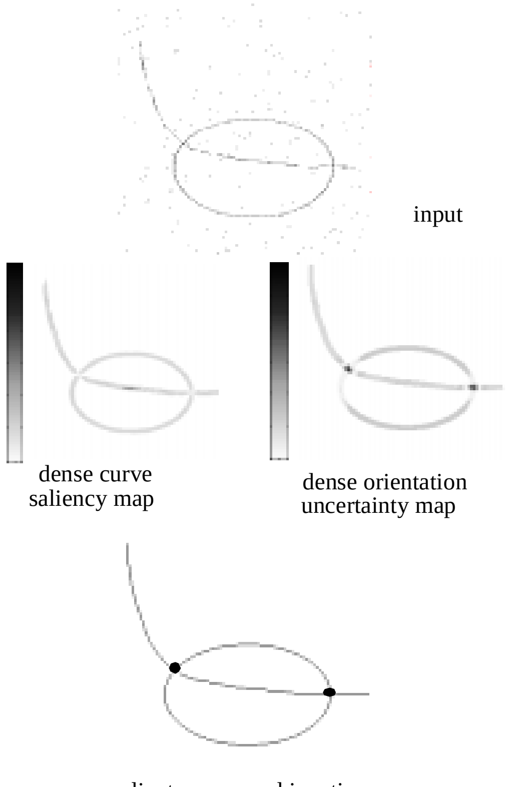

input
dense curve saliency map
dense orientation uncertainty map
salient curves and junctions

Figure 5. Inference of curves and junctions in 2-D from mixed input

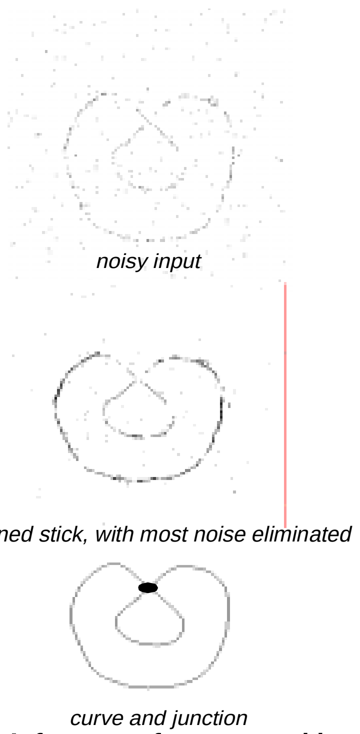


Figure 6. Inference of curves and junctions from noisy input

4.2 3-D

We generate a pipe-like object and randomly sample points on it. Then, we add 100% of random noise points to the data (i.e., only 1 point out of 2 is good). Another example is two linked tori, with 50% noise. In both cases, a first pass of tensor voting is performed to infer normal information. Then, a second pass of voting is performed which produces the dense extrapolation required for surface and curve extraction. The resulting surface and junction curve inferred are shown in Figure 7. Note the noisy-ness and the topological difficulty associated with this example. In Figure 8, we show an application to dental restoration. Note that we can infer not only the shape of the tooth, but also the grooves and the preparation line, from noisy laser data.

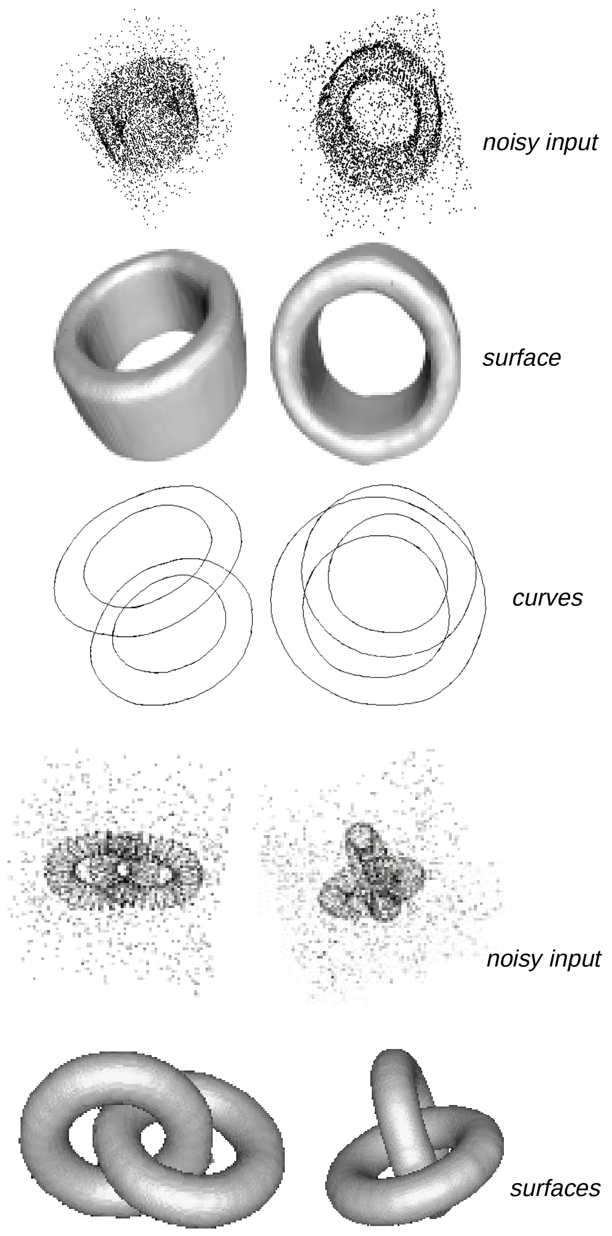


Figure 7. Pipe and two linked tori

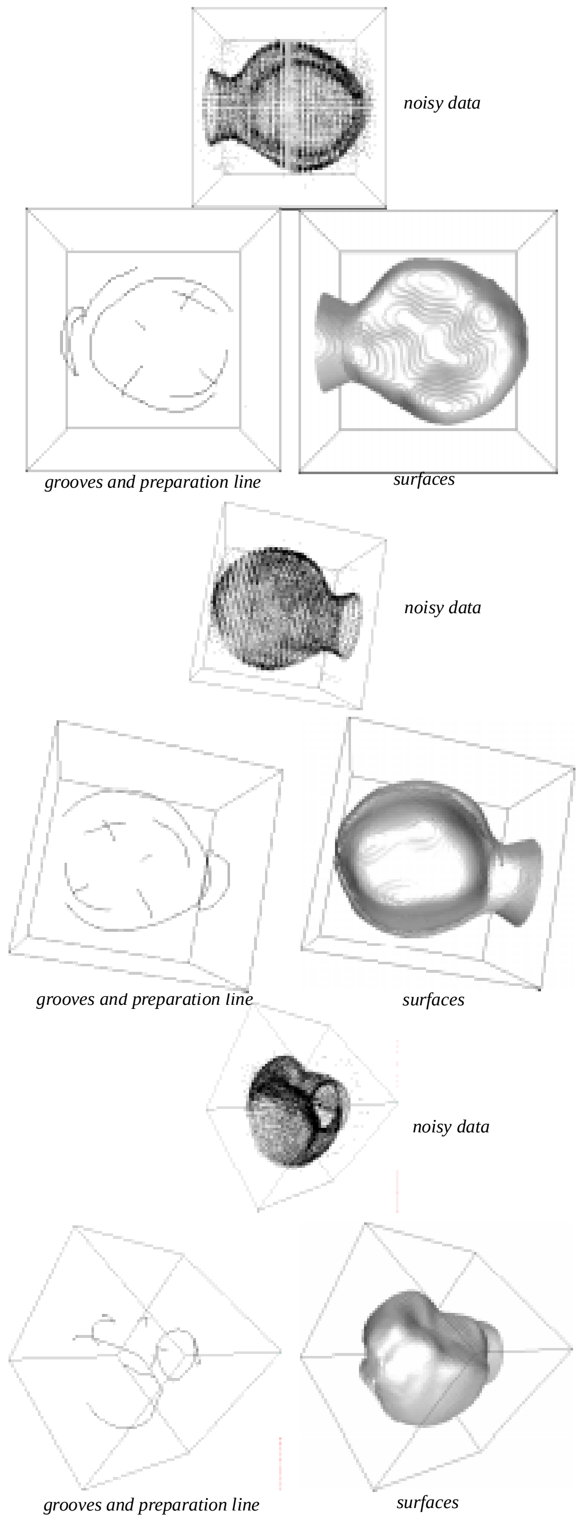

noisy data
grooves and preparation line
surfaces
noisy data
grooves and preparation line
surfaces
noisy data
grooves and preparation line
surfaces

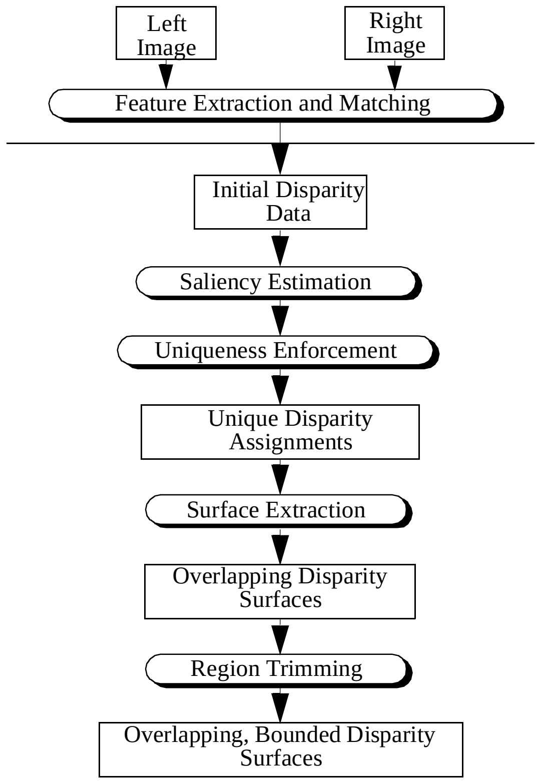
Figure 8. Dental restoration (3 views)

5 Applications to Vision and Visualization

We now show how it is possible to apply our methodology to individual problem instances in early vision and also visualization, by incorporating problem-instance specific constraints.

5.1 Stereo

The derivation of scene description from binocular



Left Image
Right Image
Feature Extraction and Matching
Initial Disparity Data
Saliency Estimation
Uniqueness Enforcement
Unique Disparity Assignments
Surface Extraction
Overlapping Disparity Surfaces
Region Trimming
Overlapping, Bounded Disparity Surfaces

Figure 9. Overview of the shape from stereo method

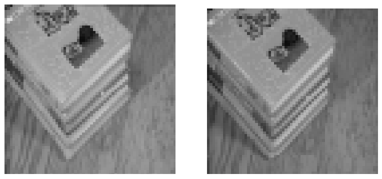
stereo images involves two processes: establishing feature correspondences across images, and reconstructing scene surfaces from the depth measurements obtained from feature correspondences. The basic constraints used in the two processes are common, namely, the uniqueness and the continuity constraints (proposed by Marr [17]). The issues needed to be addressed in both cases are identical, namely, the presence of noise, indistinct image features, surface discontinuities, and half occlusions [8]. Despite the similarities, these two processes are traditionally implemented sequentially. Instead, Hoff and Ahuja [11] have argued that the steps of matching and surface reconstruction should be treated simultaneously. In this section, we present an algorithm which approaches the shape from stereo problem from the same perspective. Given a calibrated stereo image pair, we derive a scene description in terms of surfaces, junctions, and region boundaries directly from local measurements of feature correspondence.

Figure 9 depicts an overview of our algorithm for inference using both binocular and monocular information. A running example is given in Figure 10 to illustrate the steps. Given a pair of binocular images (Figure 10(a)), we obtain an initial set of disparity assignment using traditional cross-correlation (Figure 10(b)). To evaluate the saliency of the disparity assignments, each match point cast votes during the tensor voting process using the surface inference voting field. The surface saliency map is computed after voting. In order to identify false matches, we keep the most salient match, and remove all other matches with lower saliency values, along each line of sight (Figure 10(c)). Salient surfaces and junctions are then extracted from the resulting saliency tensor field (Figure 10(d)). In order to rectify the poorly located region boundaries due to the weakness of the correlation process, we use the monocular edge information to trim down the inferred surface. For correct localization of the surface boundary, occluded points of salient surfaces are also allowed to stay in the data cluster. A tensor voting based region inference procedure is then applied to infer bounded surfaces (Figure 10(e)). A detailed description can be found in [18]. Another example is shown in Figure 11.

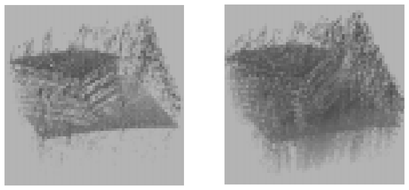
5.2 Motion

In this section, we extend our tensor representation and voting scheme to address the problem of motion flow estimation for a scene with multiple moving objects, observed from a possibly moving camera. We take as input a (possibly sparse) noisy velocity field, as obtained from local matching, and produce as output a set of motion boundaries, a dense velocity field within each boundary, and identify pixels with different velocities in overlapping *layers*. For a fixed observer, these overlapping layers capture occlusion information. For a moving observer, further processing is required to segment independent objects and infer 3-D structure. One of the advantages of our framework over previous approaches, is that we do not need to generate layers by iteratively fitting data to a set of predefined parameters. Instead, we find boundaries first, then infer regions and specifically determine occlusion overlap relationships. We use the framework of tensors to represent velocity information, together with saliency (confidence), and uncertainty. Communication between sites is performed by convolution-like tensor voting. A detailed description can be found in [18].

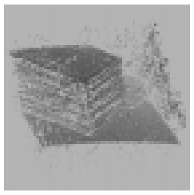
Figure 12 illustrates the steps of our method. The input is a field of velocity vectors, derived here via a three-frame maximum cross-correlation technique. We then generate a dense tensor velocity field, which encodes not only velocity, but also estimates of confidence (saliency) and uncertainty. We then extract discontinuities from this field, which are found as locations of maximal velocity uncertainty using the tensor voting formalism. Interpreting these uncertain velocity locations as local estimates of boundaries of regions, tensor voting is used again to both align tangents along these



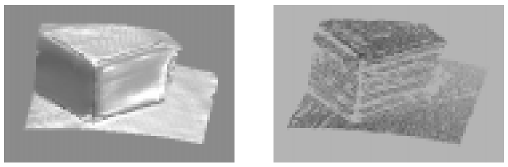
(a) input images



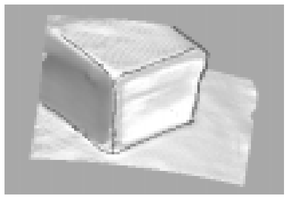
(b) initial point (both) and line (right only) correspondences



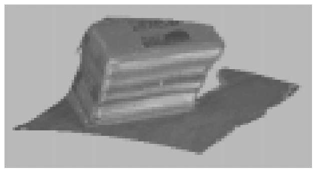
(c) unique disparity assignments



l) inferred surfaces and junctions, and the associated disparity assignments



(e) inferred surfaces, surface discontinuities and region boundaries



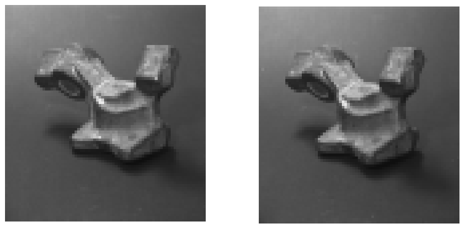
(f) a rectified, texture mapped view of the recovered scene description

Figure 10. Shape from stereo for a real scene

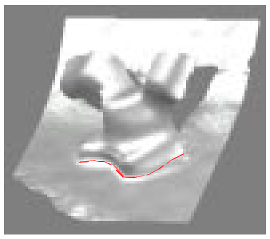
boundaries, and to join these tangents into region boundaries.

Having segmented the motion field, tensor voting is used again between pixels not separated by boundaries to accurately estimate the velocities at the borders of these objects (which are inherently uncertain in the presence of occlusion).

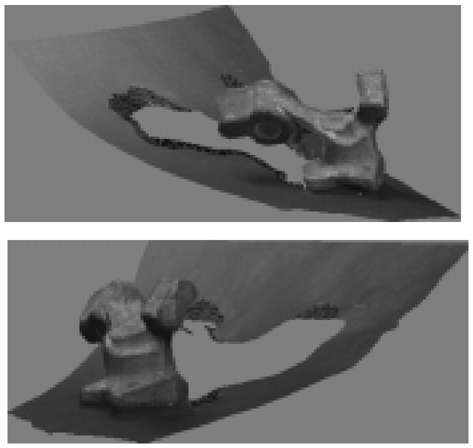
With coherent velocities at the borders of these objects, a *local* representation of occlusion is found by



(a) input images



(b) inferred surfaces and junctions in the disparity space.



(c) two views of the rectified, texture mapped view of the recovered scene description

Figure 11. Shape from stereo for a Renault par

determining which region's velocity field dominates in both future and past frames. From this analysis, the locations of pixels with multiple velocities are determined.

5.3 Flow visualization

This example demonstrates the detection, extraction, and visualization of shock waves, given a flow field such as a velocity field in a viscous medium. Figure 13 depicts the experimental set-up of a *Blunt Fin* (see Hung and Bunning [12]) and the velocity field. Note that the display of the whole velocity field is very difficult to understand. The experimental configuration is as follows: air flows over a flat plate with a blunt fin rising from the plate. The free stream flow direction is parallel to the plate and the flat part of the fin, i.e., entirely in the x component direction [67]. Figure 14 depicts four slices of the velocity field. Note the abrupt change in the flow pattern that creates a shock, as the fluid hits the blunt end of the fin.

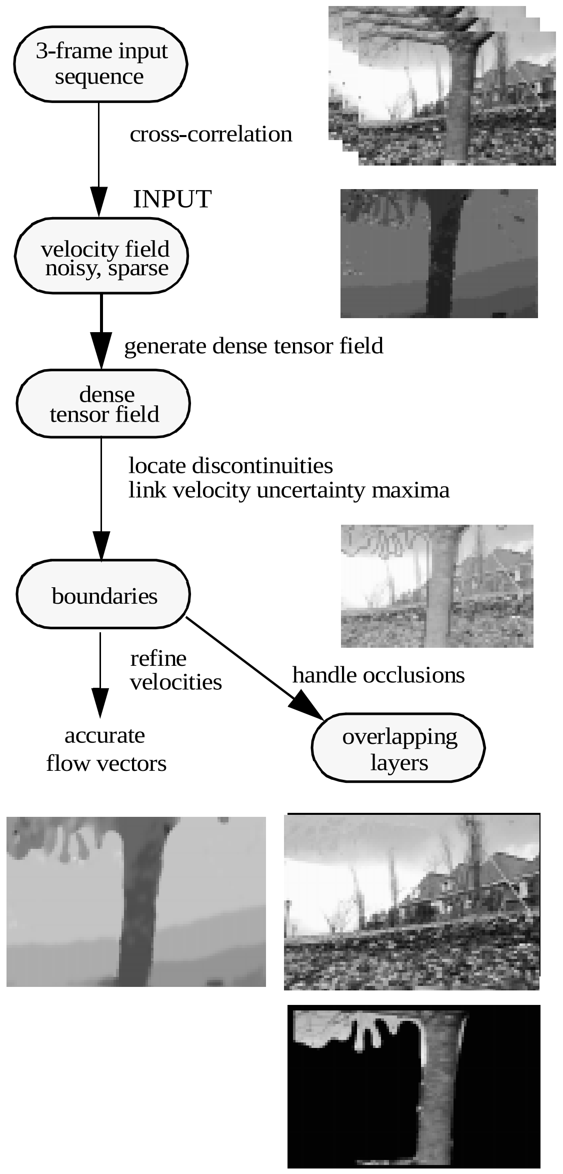


Figure 12. Overview of the algorithm for motion flow estimation

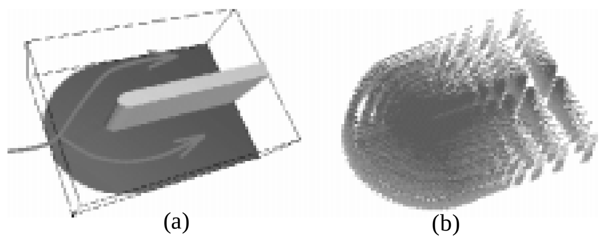


Figure 13. Input data for flow visualization

The presence of shock wave is characterized by local maxima of density gradient, c.f. Pagendarm and Walter [19], which is in coherence with the definition of surface extremality, and thus are extracted as extremal

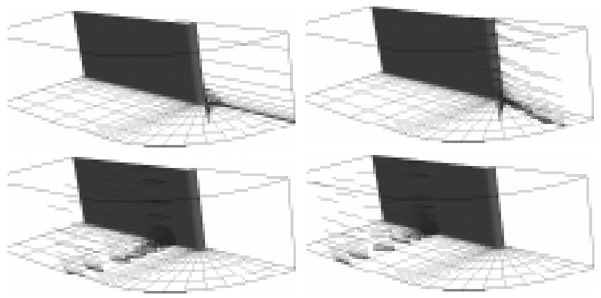


Figure 14. Velocity field of the Blunt fin

surfaces. First, we compute the density field (the left column of Figure 15 shows two views of different slices of the density field). Raw local density maxima are extracted in the density field, which results in a *sparse* set of points. Also, the original data set is sampled on a *curvilinear* grid. Therefore, a tensor voting pass is needed. Each site in the resulting dense field holds a 2-tuple $(s, \hat{\boldsymbol{n}})$ where s is the magnitude of density and $\hat{\boldsymbol{n}}$ denotes the estimated normal. The dense field is input to the extremal surface algorithm. The resulting extremal surface, corresponding to the shock wave known as a "λ-shock" [12] due to its branching structure and shape, is faithfully and explicitly extracted (c.f. [19]) and shown in the right column of Figure 15.

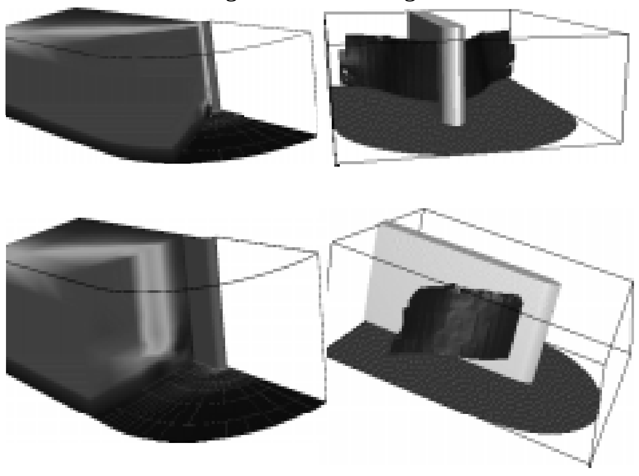


Figure 15. Density field and the extracted λ-shock

6 Conclusion and future work

In this paper, we have presented a complete formalism, tensor voting, to address the problem of salient structure inference from 2-D and 3-D data which can be contaminated with noise. Our methodology is grounded on tensor calculus for data representation, and tensor voting for data communication. Our method is non-iterative, and the only free parameter is the scale, which in fact is a property of human visual perception. The basic formalism has been extended to solve other problems as well, such as shape from stereo, motion segmentation, and flow visualization, and we have obtained very encouraging results. Our ongoing work focus on the generalization of the basic formalism to higher dimensions [27], and also the use of higher order description [28].

References

[1] N. Ahuja and M. Tuceryan, *Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component Gestalt*, Computer Vision, Graphics, and Image Processing, vol. 48, 1989, pp. 304-356.

[2] A. Blake and A. Zisserman, *Invariant Surface Reconstruction using Weak Continuity Constraints*, Proc. Conf. Computer Vision and Pattern Recognition, Miami Beach, FL, 1986, pp. 62-67.

[3] J.D. Boissonnat, *Representation of Objects by Triangulating Points in 3-D space*, Proc. 6th ICPR, pp. 830-832, 1982.

[4] J. Dolan and R. Weiss, “Perceptual Grouping of Curved Lines”, *Proc. Image Understanding Workshop*, 1989, pp. 1135-1145.

[5] H.Edelsbrunner and E.Mucke, *Three-dimensional alpha shapes*, ACM Transactions on Graphics, pp.43-72, 13, 1994.

[6] P. Fua and P. Sander, *Segmenting Unstructured 3D Points into surfaces*, ECCV 92, Santa Margherita Ligure, Italy, May 1992, pp. 676- 680.

[7] D. Geiger, K. Kumaran, and L. Parida, *Visual Organization for Figure/Ground Separation*, Proc. CVPR, San Francisco, CA, 1996, pp. 155-160.

[8] G.H. Granlund and Knutsson, *Signal Processing for Computer Vision*, Kluwer Academic Publishers, 1995.

[9] G. Guy and G. Medioni, “Inferring Global Perceptual Contours from Local Features”, Int. J. Computer Vision, vol 20, no. 1/2, 1996, pp. 113-133.

[10] G. Guy and G. Medioni, “Inference of Surfaces, 3D Curves, and Junctions from Sparse, Noisy, 3-D Data”, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no.11, Nov 1997, pp. 1265-1277.

[11] W. Hoff and N. Ahuja, “Surfaces from Stereo: Integrating Feature Matching, Disparity Estimation, and Contour Detection”, *IEEE PAMI*, vol. 11, no. 2, Feb 1989, pp. 121-136.

[12] C.M. Hung and P.G. Bunning, “Simulation of Blunt-Fin Induced Shock Wave and Turbulent Boundary Layer Separation,” *AIAA Aero. Sci. Conf.* (Reno, NV), January 1984.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, *Surface Reconstruction from Unorganized Points*, Computer Graphics, 26, pp. 71-78, 1992.

[14] M. Kass, A. Witkin, and D. Terzopoulos, *Snakes: Active Contour Models*, International Journal of Computer Vision, January 1988, pp.321-331.

[15] M.S. Lee and G. Medioni, *Inferring Segmented Surface Description from Stereo Data*, Proc.Conference on Computer Vision and Pattern Recognition, Santa Barbara, CA, June, 1998.

[16] W.E. Lorensen and H.E. Cline, “Marching Cubes: A High Resolution 3-D Surface Reconstruction Algorithm”, *Computer Graphics*, vol. 21, no. 4, 1987.

[17] D. Marr, *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, W.H. Freeman and Co., San Francisco, 1982.

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

[19] H.-G. Pagendarm and Birgit Walter, "Feature Detection from Vector Quantities in a Numerically Simulated Hypersonic Flow Field in Combination with Experimental Flow Visualization," in Proc. *IEEE Vis ’94 Conf.*, 1994.

[20] P. Parent and S.W. Zucker, “Trace Inference, Curvature Consistency, and Curve Detection”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no.8, 1989, pp. 823-839.

[21] T. Poggio and F. Girosi, *A theory of Networks for Learning*, Science, 247, pp. 978-982, 1990.

[22] S. Sarkar and K.L. Boyer, *Integration, Inference, and Management of Spatial Information Using Bayesian Networks: Perceptual Organization*, PAMI, vol. 15, no. 3, 1993, pp. 256-274.

[23] A. Sha’ashua and S. Ullman, “Structural Saliency: the Detection of Globally Salient Structures using a Locally Connected Network”, *Proc. Int. Conf. on Computer Vision*, 1988, pp.321-327.

[24] J. Shi and J. Malik, *Normalized Cuts and Image Segmentation*, Proc. Computer Vision and Pattern Recognition, Jun. 1997, Puerto Rico, pp. 731-737.

[25] R. Szeliski, D. Tonnesen, and D. Terzopoulos, *Modeling Surfaces of Arbitrary Topology with Dynamic Particles*, Proc. CVPR, 1993, New York, NY, pp. 82-85.

[26] C.K. Tang and G. Medioni, “Integrated Surface, Curve and Junction Inference from Sparse 3-D Data Sets”, *Proc. ICCV 98*, India, Jan. 1998.

[27] C.K. Tang, G. Medioni and M.S. Lee, “Epipolar Geometry Estimation by Tensor Voting in 8D,” *Proc. ICCV 99*, Greece, Sep 1999.

[28] C.K. Tang and G. Medioni, “Robust Estimation of Curvature Information for Shape Description,” *Proc. ICCV 99*.

[29] D. Terzopoulos, A. Witkin, and M. Kass, *Constraints on deformable models: Recovering 3D Shape and Nonrigid Motion*, *Artificial Intelligence*, Vol. 36, 1988, pp. 91-123.

[30] L.R. Williams and D. Jacobs, *Stochastic Completion Fields: A Neural Model of Illusory Contour Shape and Salience*, Proc. 5th ICCV, Cambridge, MA, Jun. 1995, pp. 408-415.

[31] K. Thornber and L.R. Williams, *Analytic Solution of Stochastic Completion Fields*, BioCyber, vol 75, 1996, pp. 141-151.

[32] H.-K. Zhao, S. Osher, B. Merriman and M. Kang, *Implicit,Nonparametric Shape Reconstruction from Unorganized Points Using A Variational Level Set Method*, UCLA Computational and Applied Mathematics Reports 98-7, February 1998 (revised February 1999).

[33] S.W. Zucker and R.A. Hummel, “Toward a Low-Level Description of Dot Clusters: Labeling Edge, Interior, and Noise Points”, *Computer Vision, Graphics, and Image Processing*, vol. 9, no.3, 1979, pp.213-234.