Inference of Integrated Surface, Curve, and Junction Descriptions From Sparse 3D Data

Chi-Keung Tang, *Student Member, IEEE,* and Gérard Medioni, *Senior Member, IEEE*

Abstract—We are interested in descriptions of 3D data sets, as obtained from stereo or a 3D digitizer. We therefore consider as input a sparse set of points, possibly associated with certain orientation information. In this paper, we address the problem of inferring integrated high-level descriptions such as surfaces, 3D curves, and junctions from a sparse point set. While the method proposed by Guy and Medioni provides excellent results for smooth structures, it only *detects* surface orientation discontinuities but does not localize them. For precise localization, we propose a noniterative cooperative algorithm in which surfaces, curves, and junctions work together: Initial estimates are computed based on the work by Guy and Medioni, where each point in the given sparse and possibly noisy point set is convolved with a predefined vector mask to produce dense saliency maps. These maps serve as input to our novel extremal surface and curve algorithms for initial surface and curve extraction. These initial features are refined and integrated by using excitatory and inhibitory fields. Consequently, intersecting surfaces (resp. curves) are fused precisely at their intersection curves (resp. junctions). Results on several synthetic as well as real data sets are presented.

Index Terms—Segmentation and feature extraction, integrated shape description, surface orientation discontinuity, surface and curve extremality.

1 Introduction

Our human visual system can perform an amazingly good job of perceiving surfaces from a set of 3D points. We not only can infer surfaces, but also segment the scene, and detect surface orientation discontinuities. For example, Fig. 1 shows a sparse set of points (with normals) sampled from a planar surface intersecting with a sphere. The circle represents the intersection contour that is *not* explicit in the data. When the data is presented to us as a sequence of projections, we ourselves have no problem in inferring such surface orientation discontinuities (i.e., the circular intersection curve) and segment the scene into two components, a spherical and a planar surface. Earlier work by Guy and Medioni [6] proposed to *detect* the presence of junctions and intersection curves from such data. While it did a good job of detection, it did not try to integrate them into a unified representation, but instead produced three independent representations: one for surfaces, one for curves, and one for junctions. It can be readily observed from their results that the surfaces inferred are only correct away from curves and junctions, and that curves and junctions are not properly localized.

Our approach is based on this work [6], where a nonlinear voting process (implemented as a convolution of input features with a predefined mask) is used to enforce perceptual constraints in order to achieve efficient segmentation and discontinuity detection. Our main contribution is to show that this voting process, when combined with our novel surface and curve extraction processes, can be readily extended to unify these three independent representations to produce an *integrated* description of surfaces, curves, and junctions.

We start by briefly reviewing some of the existing work including [6] on this problem, then motivate and describe our approach, and finally show results on complex synthetic and real data. A compact version with an incomplete account of Sections 5.2 and 5.3 (due to space limit) of this paper has appeared in [19]. We have improved our implementation and also its time and space complexities reported in [19]. The present coverage presents our work in complete detail, while readers can regard our delicate surface and curve extraction processes as plug-in components.

2 Previous Work

Much work has been done in surface fitting to clouds of points. The majority of work uses the *deformable model* approach (first proposed by Kass et al. in [11] and [22]) which attempts to deform an initial shape using energy minimization so that the deformed shape fits a set of points. Boult and Kender [3] addressed the sparse data problem specifically and demonstrated a method using minimization over Hilbert spaces. Poggio and Girosi [16] formulated the single-surface approximation problem as a network learning problem. Blake and Zisserman [4] addressed similar problems dealing explicitly with discontinuities. Fua and Sander [5] proposed an algorithm to describe surfaces from a set of points using local properties. Szeliski et al. [18] formulated

• *The authors are with the Institute for Robotics and Intelligent Systems, University of Southern California, CA 90089-0273.*
E-mail: {chitang, medioni}@iris.usc.edu.

Manuscript received 24 Feb. 1998; revised 9 Sept. 1998. Recommended for acceptance by R. Szeliski.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107419.

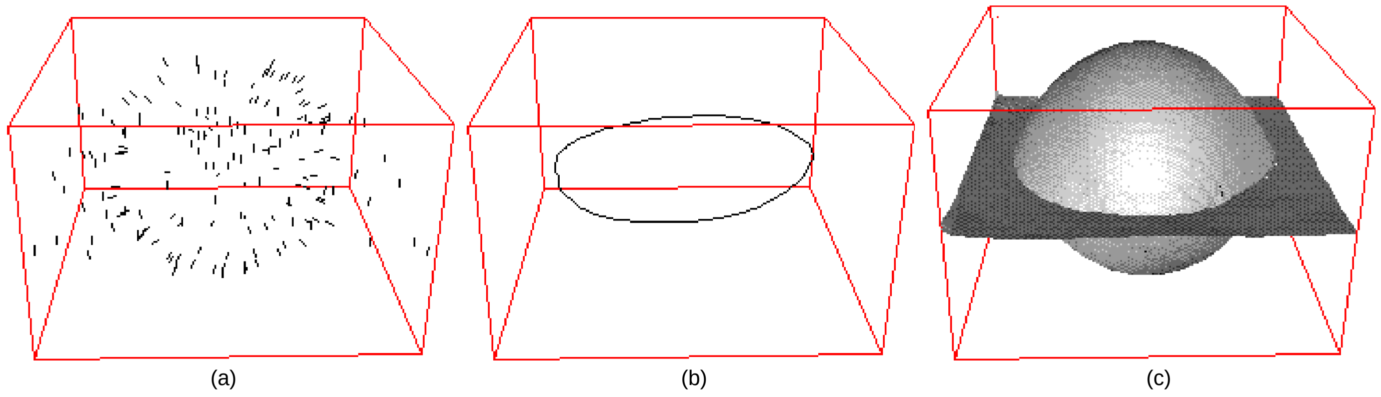


Fig. 1. Inferring integrated high-level description. (a) Input sparse data points with normals. (b) Surface orientation discontinuity is localized as an intersection curve. (c) Output surface with the discontinuity preserved.

the problem as a physically-based dynamic process that makes use of local properties from each data point. A *single* object of unrestricted topology can be handled using their method. In [7], [13], an initial surface is made to fit the data by minimizing energy function, which is derived from the smoothness and proximity constraints. *Physics-based* approaches proposed by Terzopoulos et al. (in [20], [21], and [24]) model the problem as a multi-particle system governed by (physics) laws and equations. The initial surface (or model), originally in equilibrium, is now subject to external forces exerted at each data point. Such forces make the system converge to another equilibrium. Hoppe et al. [9], [10] and Boissonnat [1] use *computational geometry* to address the problem by treating the data as vertices of a graph and constructing edges based on local properties. However, in [9], creases were not located and blurred out. By optimizing the mesh [10] in a sequel of their work, they were able to detect sharp features in the underlying surface and segment the optimized mesh into smooth components. Our work uses a voxel representation by quantizing the (sparse and noisy) input and produces surface meshes. This voxel approach is also used by Hilton et al. [8], Wheeler et al. [25], and Roth and Wibowo [17]. Their methods deal with dense and accurate 3D data.

3 Method Proposed by Guy and Medioni

Most of the above methods are computationally expensive since many iterations may be required. Moreover, they have one or more of the following limitations:

- multiple objects cannot be handled;
- only relatively simple topologies can be described;
- surface boundaries and orientation discontinuities are usually smoothed out;
- do not work in the presence of noise.

Guy and Medioni [6] have recently proposed a method to attack these problems. Since our work is a direct extension, we shall summarize it in more detail to serve as background as well as a source of terminology that will be used throughout this paper.

Fig. 2 shows the major steps of the algorithm. Each input site in the sparse data set is quantized in a 3D array. A surface normal or tangent must be associated with each point. A preprocessing step to estimate surface normals is required when they are unavailable. The preprocessing as well as the vote accumulation is implemented as convolution with various vector *fields* (Fig. 3).

3.1 Fields

There are three types of 3D vector fields proposed in [6]: the *3D point field* (*P-field*), *curve segment field* (*C-field*), and *Diabolo field*[1] (*D-field*). The first two are used in preprocessing for normal estimation and the third one is used for surface inference. In all cases, the length of each vector is inversely proportional to the distance from O (for the D-field also the curvature of the underlying circular arc connecting O and P). A Gaussian decay function is used for that purpose.

P-field. Without any a priori assumption, given a point P and an origin O, the most likely surface passing through O and P is a family of planes containing these two points, represented by a single vector $\overline{OP}$ (Fig. 3a).

C-field. Without any a priori assumption, given a point P in space with the associated tangent vector lying at the origin, the most likely surface is the family of planes

1. This vector field is named as the "Diabolo field" because of its resemblance to a juggler's Diabolo [6].

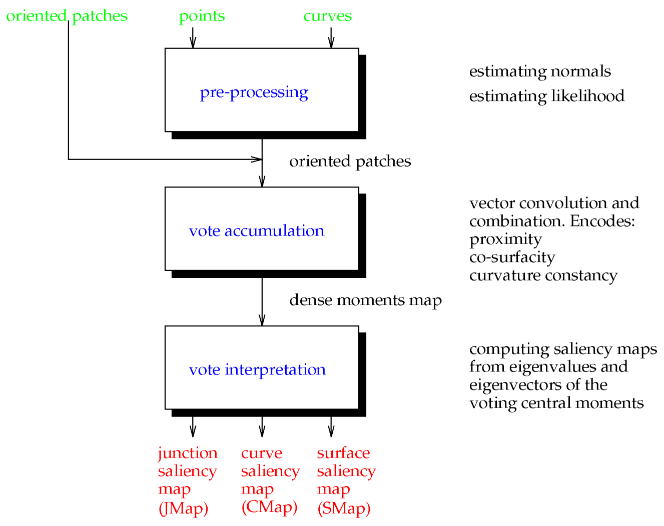

oriented patches
points
curves
pre-processing
estimating normals
estimating likelihood
oriented patches
vote accumulation
vector convolution and combination. Encodes: proximity co-surfacity curvature constancy
dense moments map
vote interpretation
computing saliency maps from eigenvalues and eigenvectors of the voting central moments
junction saliency map (JMap)
curve saliency map (CMap)
surface saliency map (SMap)

Fig. 2. Flowchart of the method proposed by Guy and Medioni.

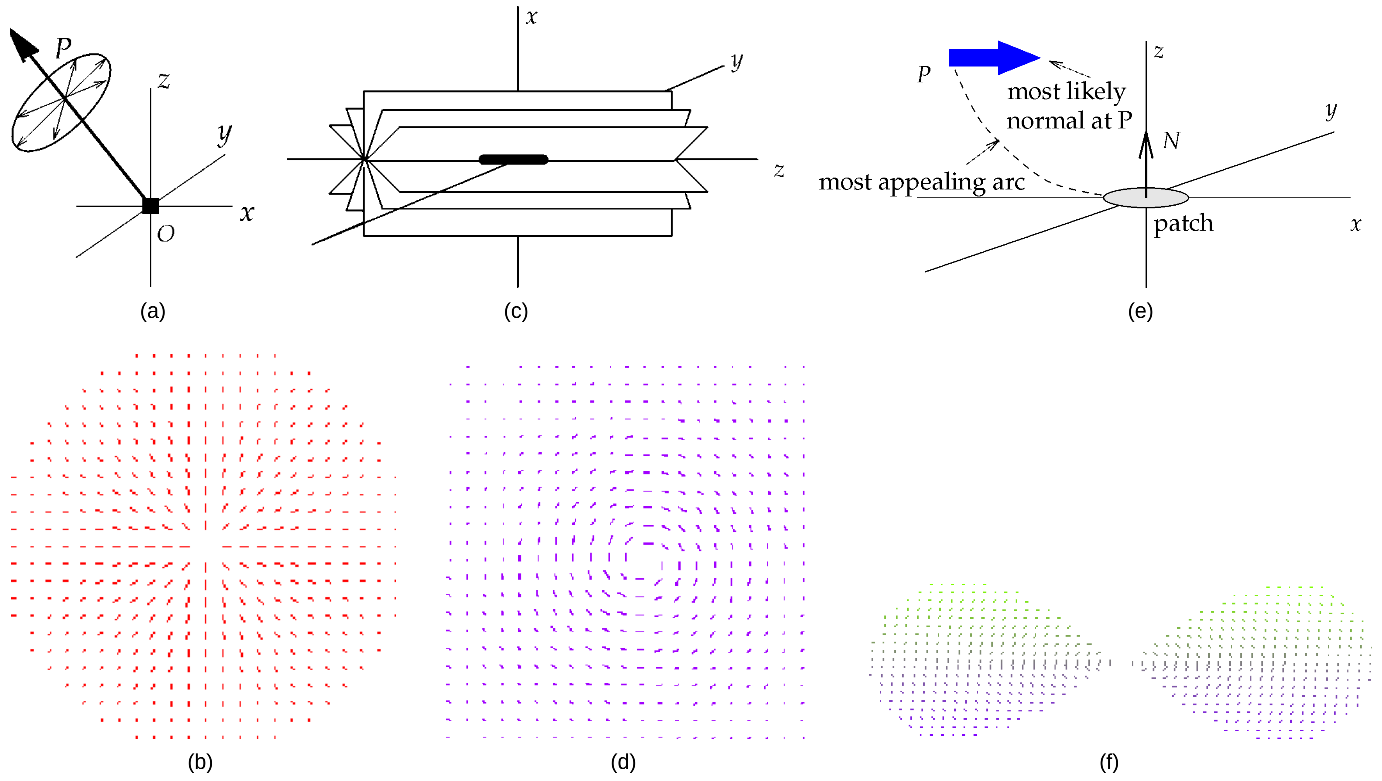


Fig. 3. (a) Given a point *P*, all normals (thin arrows) at point *P* are equally likely. We choose to represent all of them with one vector (thick arrow). (b) Cross section of P-field at $y = 0$. (c) Given a curve segment, all planes containing this segment are equally likely. (d) Cross section of C-field at $y = 0$. (e) Given a patch and point *P*, the circular continuation is most likely. (f) Cross section of D-field at $y = 0$.

containing that tangent vector (Fig. 3c). The normal vectors to the family of planes are collected as the C-field (Fig. 3d).

D-field. Given a normal vector N at origin O and a point P (Fig. 3e), without any a priori information, there is no reason to postulate a more complex surface than a circular continuation. Therefore, we claim that the "most likely" surface through P will be the circular continuation between O and P, because it keeps the curvature constant. The "most likely" normal is normal to that arc at P. The collection of such most likely normal vectors comprises the *D-field* (Fig. 3f).

3.2 Vector Voting and Saliency Maps

We firstly describe the basic case when surface normals are available as input. The output (and input to our cooperative algorithm described next) is three independent *dense saliency maps* (defined shortly). Computing saliency maps is done by voting, which is realized by *convolving* each input normal vector with the D-field. The resulting map is a collection of fields, each oriented along the input normal vector. Each site accumulates the "votes" for its own preferred orientation and strength from every other input in the volume. The contributions to a voxel are treated as vector weights, and we compute the central moments of the resulting system. This is equivalent to computing a 3D ellipsoid having the same moments and principal axes. Such a physical model acts as an approximation to a majority vote which gives both the preferred direction and a measure of the agreement. For each voxel (x, y, z) in the *entire* 3D array, we define the accumulated vote, O_{xyz}, as a 3×3 variance-covariance matrix. We decompose this matrix of central moments into the corresponding eigensystem,

$$O_{xyz} = \begin{bmatrix} V_{min} & V_{mid} & V_{max} \end{bmatrix} \begin{bmatrix} \lambda_{min} & 0 & 0 \\ 0 & \lambda_{mid} & 0 \\ 0 & 0 & \lambda_{max} \end{bmatrix} \begin{bmatrix} V_{min}^T \\ V_{mid}^T \\ V_{max}^T \end{bmatrix} \qquad (1)$$

as expressed in (1), where $\lambda_{min} \le \lambda_{mid} \le \lambda_{max}$ represent the three sorted eigenvalues, and the Vs denote the corresponding eigenvectors of the system. Such decomposition will always yield real, nonnegative eigenvalues since the matrix is real, symmetric, positive, and semidefinite.

The three eigenvectors correspond to the three principal directions of an ellipsoid in 3D, while the eigenvalues describe the strength and agreement measures of the 3D votes. On a smooth surface, the votes produce high agreement around one direction, so $\lambda_{max} \gg \lambda_{mid}, \lambda_{min}$ (Fig. 4a). Along the curve bounding two surfaces, two of the eigenvalues are high, and one small, leading to $\lambda_{mid} \gg \lambda_{min}$ (Fig. 4b). Finally, at a junction of two or more curves, all three values are similar (Fig. 4c). Thus three voxel maps defining the surface, curve, and junction *saliencies*, respectively are proposed. Each voxel of these maps has a two-tuple $(s, \overline{v})$, where s is a scalar measuring *feature saliency* and $\overline{v}$ is a unit vector indicating *direction*:

- surface map (*SMap*): $s = \lambda_{max} - \lambda_{mid}$, and $\overline{v} = V_{max}$ indicating the normal direction.
- curve map (*CMap*): $s = \lambda_{mid} - \lambda_{min}$, and $\overline{v} = V_{min}$ indicating the tangent direction.
- junction map (*JMap*): $s = \lambda_{min}$, with $\overline{v}$ arbitrary.

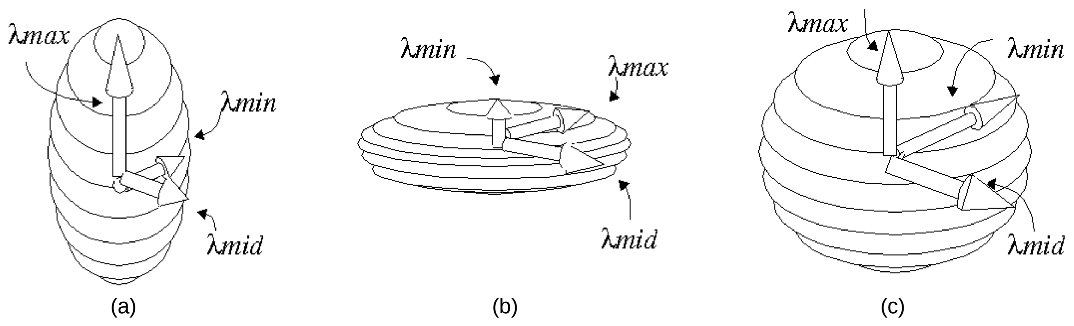


Fig. 4. The three important voting ellipsoids.

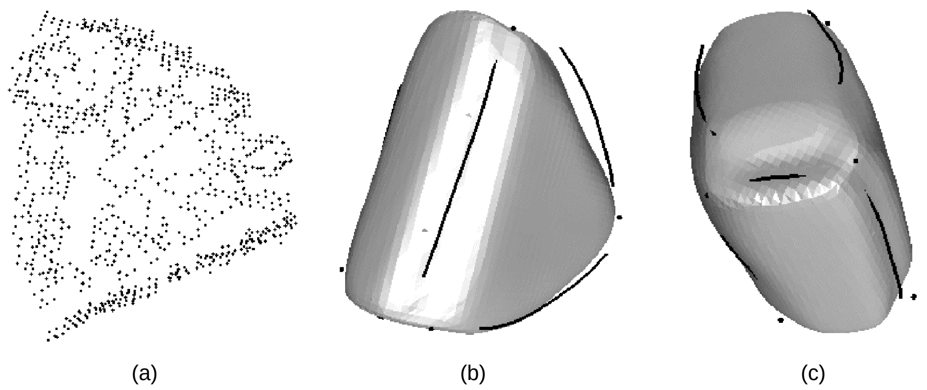


Fig. 5. (a) Input data obtained from a digitized surface sweep on a triangular wedge. (b) and (c) Two views of the reconstructed surfaces, 3D curves and junctions. Surface (resp. curve) orientation discontinuities, though detected, are not well localized because SMap (resp. CMap) is used alone for surface (resp. curve) tracing.

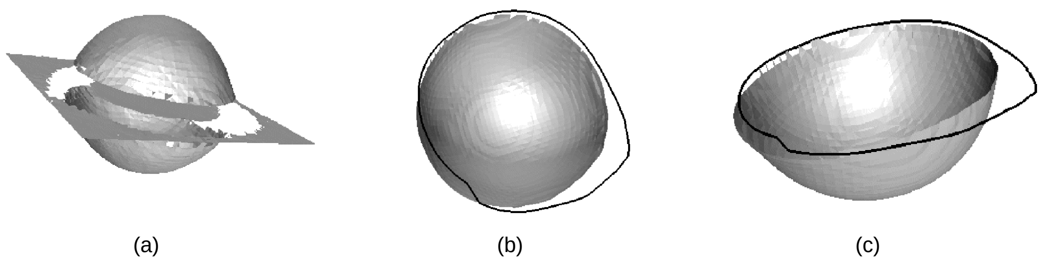


Fig. 6. Incorrect curve may be obtained by considering the CMap alone. (a) Gaps are produced around region of low surface saliency. (b) and (c) Two views of the incorrect curve owing to data sparsity.

3.3 Preprocessing for the Nonoriented Cases

When we have points or oriented curve elements as input, we preprocess them to infer a normal. This is achieved again by convolving each input site with the appropriate vector kernel (P-field or C-field), and interpreting the resulting votes, but only at the original input sites. As a result of this step, each input site now holds a normal, obtained as the eigenvector V_{max} corresponding to λ_{max}.

4 Motivation and Overall Strategy

An integrated high level description requires that surface orientation discontinuities be explicitly preserved. However, as readily seen from [6], generating surfaces (resp. curves and junctions) using the SMap (resp. CMap and JMap) *independently* does not guarantee such precise discontinuities. Discontinuities, though detected, may be:

- Smoothed out. For example, although a salient curve may be *detected* in CMap, it is still possible that the surface saliency gradient across the corresponding voxels in SMap varies smoothly and thus a smooth surface will be traced if the SMap is *alone* considered, and surfaces, curves, and junctions are not coherently integrated (Fig. 5).
- Left "undecided." Voxels around surface orientation discontinuities have a low surface saliency and thus no surface will be produced, creating a gap (Fig. 6a).
- Incorrect. Because of data sparsity, using the CMap *alone* to trace a curve may produce incorrect result. A sphere intersecting with a plane should give a circle. However, if we use the CMap *alone* in this case to infer the intersection contour, it will not be circular (as shown in Fig. 6 where one of the hemispherical surfaces is also shown).

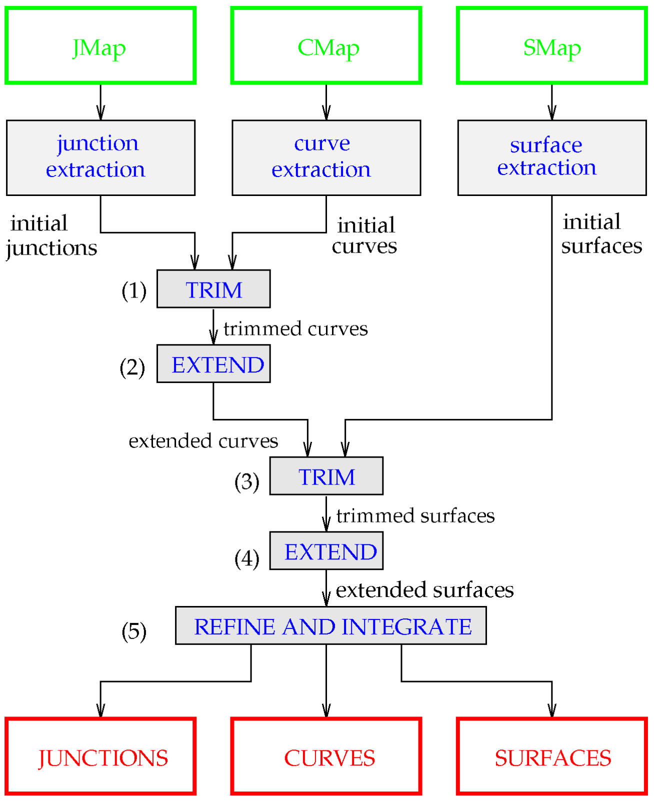


Fig. 7. Overview of cooperative algorithm.

We have illustrated through examples the limitations of noncooperative feature inference. In this paper, we show (constructively by proposing an algorithm) that while the original work by Guy and Medioni [6] does not handle an integrated description, their voting approach can be extended cleanly into a *cooperative* framework such that surfaces, curves, and junctions can be inferred cooperatively. The underlying idea is two-fold:

- Extending the use of the D-, C-, and P-fields in hybrid voting (using different fields in a single voting pass) to infer surface/curve orientation discontinuities.
- Making use of our novel extremal surface/curve extraction processes to be described for initial estimate generation and subsequent refinement.

Our overall strategy is as follows: For preserving precise surface orientation discontinuity, we treat the curve as a *surface inhibitor*: The curve will not be smoothed out while a surface is being traced using the SMap. Once an explicit initial surface estimate is obtained, we treat the very same curve as *exciter* for computing precise discontinuity. A similar rationale applies to curve orientation discontinuity. We show in Fig. 7 the major steps of the cooperative algorithm, and illustrate them in Fig. 8 using one face of our triangular wedge (Fig. 5) as a running example:

1) *Curve trimming by inhibitory junctions.* Initial junctions are convolved with a *curve inhibitory field* so that the detected discontinuity (indicated by JMap) will not be smoothed out when a developing curve is evolving (Fig. 8a).
2) *Curve extension toward excitatory curves.* Initial junctions and curve are convolved with *curve excitatory fields* so that the curve obtained in (1) are brought to intersect with the junctions (Fig. 8b).
3) *Surface trimming by inhibitory curves.* Extended curves obtained in (2) are convolved with a *surface inhibitory field* so that the detected discontinuity (indicated by CMap) will not be smoothed out when a developing surface is evolving (Fig. 8c).
4) *Surface extension toward excitatory curves.* The extended curve and the trimmed surface are convolved with *surface excitatory fields* such that latter can be naturally extended to hit the curve (Fig. 8d).
5) *Final curves and junctions from surfaces.* The set of surface boundaries obtained by extended surface intersection produces a set of refined curves and junctions which are coherently integrated and localized (Fig. 8e).

We want to emphasize here that our cooperative approach is *not* iterative (though each step makes use of results produced in previous step(s)). Also, the geometric locations of all junctions, 3D curves, and surfaces may change considerably after the cooperation process.

5 High Level Feature Extraction

Initial junction, curve, and surface estimates are generated by extracting local saliency extrema in the JMap, CMap, and SMap. However, except for junctions, raw extrema extraction in CMap or SMap only produces a *thick* set of points, not a curve or a surface. In order for the subsequent cooperation process to take place, we need a curve be represented by a set of connected poly-line segments, and a surface represented by a hole-free triangulation. To this end, we develop *extremal curve and surface algorithms* to obtain such explicit representations.

Recall that every voxel of these saliency map holds a two-tuple $(s, \overline{v})$ where s is the saliency and $\overline{v}$ indicating tangent or normal directions.

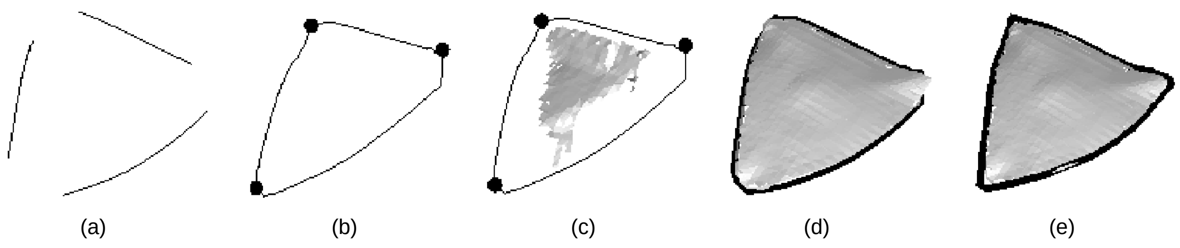


Fig. 8. Illustration of overall strategy.

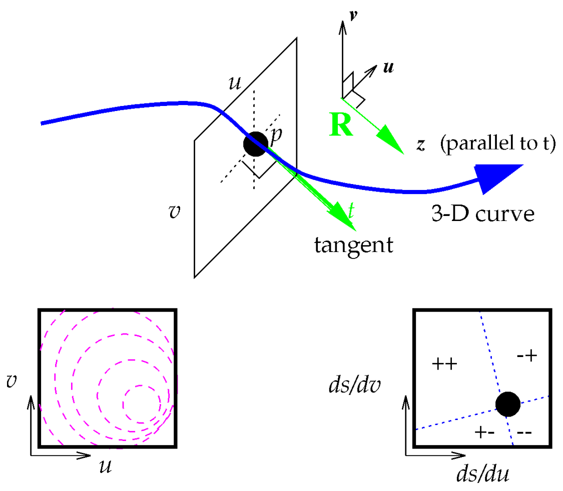


Fig. 9. Curve saliency is projected onto a plane perpendicular to the tangent. When a change in signs occurs in the derivatives in both *u* and *v*, a curve passes through the point.

5.1 Maximal Junctions

The 3D junctions are isolated points, by definition, so it is straightforward to extract them as local maxima of the λ_{min} values.

5.2 Extremal Curves

Each voxel in the CMap holds a two-tuple $(s, \bar{t})$, where the $s = \lambda_{mid} - \lambda_{min}$ is curve saliency and $\bar{t} = V_{min}$ indicates direction. Suppose the CMap is continuous, in which $(s, \bar{t})$ is defined for every point *p* in 3D space. A point *p* with $(s, \bar{t})$ is on an *extremal curve* if any displacement from *p* on the plane normal to $\bar{t}$ will result in a lower *s* value, i.e.,

$$\frac{ds}{du} = \frac{ds}{dv} = 0, \tag{2}$$

where *u* and *v* define the plane normal to $\bar{t}$ at *p* (Fig. 9).

This definition therefore involves the detection of zero-crossing in the *u*-*v* plane normal to $\bar{t}$. To do this, we introduce the gradient vector $\bar{g}$ as

$$\bar{g} = \begin{bmatrix} \frac{ds}{dx} & \frac{ds}{dy} & \frac{ds}{dz} \end{bmatrix}^T. \tag{3}$$

Define $\bar{q} = \mathbf{R}(\bar{t} \times \bar{g})$, where $\mathbf{R}$ defines a rotation that aligns with the *u*-*v* plane. By construction, $\bar{q}$ is the projection of $\bar{g}$ onto the plane normal to $\bar{t}$. Therefore, an extremal curve is the locus of points for which $\bar{q} = \bar{0}$.

In implementation, we can define the corresponding discrete version of $\bar{g}$ and $\bar{q}$, i.e., such that

$$\overline{g_{i,j,k}} = \begin{bmatrix} s_{i+1,j,k} - s_{i,j,k} \\ s_{i,j+1,k} - s_{i,j,k} \\ s_{i,j,k+1} - s_{i,j,k} \end{bmatrix} \tag{4}$$

$$\overline{q_{i,j,k}} = \mathbf{R}\left(\overline{t_{i,j,k}} \times \overline{g_{i,j,k}}\right). \tag{5}$$

Therefore, the set of all $\left\{\overline{q_{i,j,k}}\right\}$ constitutes a *vector* array, which can be processed by the Marching Cubes algorithm with our novel adaptation (`SingleSubVoxelCMarch`) to be described in the Appendix.

So the overall extremal curve algorithm picks the seed voxel with $(s, \bar{t})$ whose *s* value is largest so far, computes the point (if any) where an extremal curve passes through by using `SingleSubVoxelCMarch` which uses the discrete version of $\bar{q}$, and aggregates the curve in direction $\bar{t}$ until the current *s* value falls below a low threshold. Denote this curve thus obtained by C_1. Then the algorithm returns to the seed and repeats the whole process above with direction $-t$. Denote the curve thus obtained by C_2. It outputs *Reverse*$(C_2) \cup C_1$ as a connected and oriented extremal curve. If there are multiple curves, then it picks the next available seed and performs the same curve aggregation process until the *s* value of the next seed falls below a high threshold.

The choices of high and low thresholds in this curve extraction process are not critical. These thresholds are used only for ordering the extraction of extremal curves by their seed's saliency, and to reject features due to noise. In other words, only the running time is affected by choosing thresholds too low, but not the result. These thresholds are chosen empirically.

5.3 Extremal Surface

Each voxel in the SMap holds a two-tuple $(s, \bar{n})$ where $s = \lambda_{max} - \lambda_{mid}$ indicating surface saliency and $\bar{v} = V_{max}$ denoting the normal direction. As before, suppose the SMap is continuous in which $(s, \bar{n})$ is defined for every point *p* in 3D space. A point is on an *extremal surface* if its saliency *s* is locally extremal along the direction of the normal, i.e.,

$$\frac{ds}{d\bar{n}} = 0. \tag{6}$$

This definition involves the detection of zero crossing on the line aligned with $\bar{n}$ (Fig. 10). We compute this by projecting $\bar{g}$ onto $\bar{n}$, i.e.,

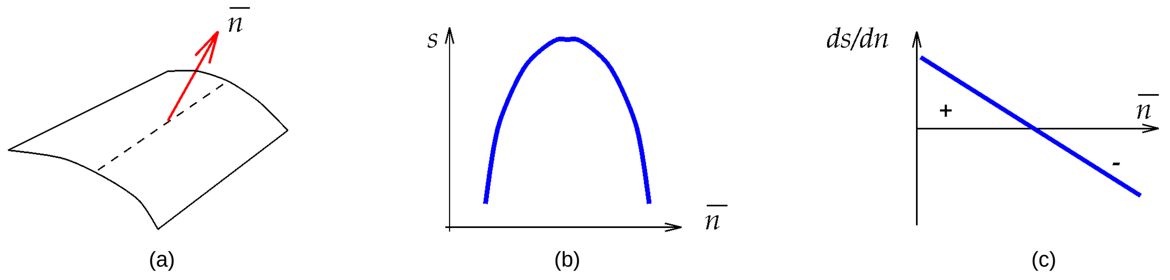


Fig. 10. (a) A normal vector (with an imaginary surface patch drawn). (b) Surface saliency along the normal. (c) The derivative of saliency.

$$q = \overline{n} \cdot \overline{g}. \qquad (7)$$

Therefore, an extremal surface is the locus of points for which $q = 0$.

As before, we can define the corresponding discrete version for q, i.e.,

$$q_{i,j,k} = \overline{n_{i,j,k}} \cdot \overline{g_{i,j,k}}. \qquad (8)$$

Therefore, the set of all $\{q_{i,j,k}\}$ constitutes a scalar field which can be processed directly by the Marching Cubes algorithm [14].

The overall algorithm picks the seed voxel whose s value is largest so far, computes a surface (if it exists) by using SingleSubVoxelMarch (to be described in the Appendix) which makes use of $\{q_{i,j,k}\}$, and aggregates the surface in its neighborhood until the current s value falls below a low threshold. If there are multiple surfaces, it then picks the next available seed and performs the same patch aggregation process until the s value of the next seed falls below a high threshold. A polygonal mesh is thus produced. Again, the choice of these thresholds are not critical.

6 Cooperative Computations and Hybrid Voting

We extend vector voting in [6] to cooperatively integrate initial junction, curve, and surface estimates generated by extremal feature extraction algorithms to obtain an integrated description. Slight modifications are needed for both feature extraction algorithms, which will be described in the following sections.

6.1 Feature Inhibitory and Excitatory Fields

In essence, the process of feature integration is to define *feature inhibitory and excitatory fields* and to use them for feature localization. We have curve and surface inhibitory fields. Curve (resp. surface) inhibitory field is an inhibition mask for inhibiting curve (resp. surface) growing as intended by its respective extremal algorithms. No curve segment (resp. surface patch) is possible in a voxel masked by a inhibitory field.

We also have excitatory fields for inferring feature extension. These fields are essentially defined by P-, C-, and D-fields for feature extension toward the detected orientation discontinuity. In particular,

- Curve excitatory fields are defined by P-field and 3D extension field (to be defined below),
- Surface excitatory fields are defined by C-field and D-field.

6.2 Curve Trimming by Inhibitory Junctions

The extremal curve extraction algorithm (Section 5.2) is modified to take not only the CMap but also the detected junction estimates (from JMap) as input:

1) The voxels in CMap corresponding to initial junctions are inhibited by a *curve inhibitory field* to protect detected curve orientation discontinuity detected in JMap. It is done simply by putting the corresponding inhibition mask over the detection junction locations (voxels). Typical size of this inhibition mask is $5 \times 5 \times 5$.
2) Curves are traced exactly as described in Section 5.2.

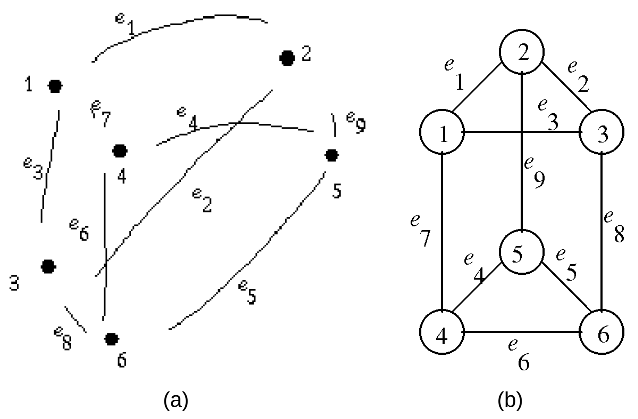


Fig. 11. (a) Initial curves and junctions. (b) An incidence graph is constructed.

Since curve growing is inhibited around a junction but not by the low threshold in the original extremal curve extraction, its orientation discontinuity will not be smoothed out, and no spurious curve features are created around a junction. This step results in a set of trimmed curves.

6.3 Curve Extension Toward Excitatory Junction

The detected junctions and trimmed curves obtained in the previous phase are used to produce an “extended” curve for which curve orientation discontinuities are preserved. We group these features by using an incidence graph (Fig. 11b), and process curve extension one by one (like divide-and-conquer strategy).

First, an incidence graph $G = (V, E)$ is constructed with E corresponding to curves and V to incident junctions (Fig. 11). This graph is created by checking the distance between every endpoint of the trimmed curves and the detected junctions to determine to which junctions the curve endpoints should be connected. By such grouping, we can avoid unnecessary and unwanted interaction among excitatory fields (defined shortly).

Then, for each curve (edge in E) and its incident junctions (vertices in V), we quantize them as input (recall that this intermediate curve is of subvoxel precision). Two excitatory fields are used to extend the curve toward detected junctions such that it will intersect the junction precisely in a single pass of voting:

1) (Excitatory 3D extension field) Each curve segment (in E) is convolved with the *3D extension field* (Fig. 12b). A 2D extension field comprises (Fig. 12a) the set of all (nonequally) likely *tangents* that will best connect two separate 2D line segments. (So its design is analogous to its 3D counterpart D-field which encodes most likely *normals*.) A 3D extension field is obtained by rotating its 2D version about its “long” axis. See Fig. 12b.
2) (Excitatory P-field) The junctions (in V) are convolved with the P-field, where each vote in P-field is increased (i.e., excitatory) in order to “attract” the curve toward the junction. Its size is related by a constant

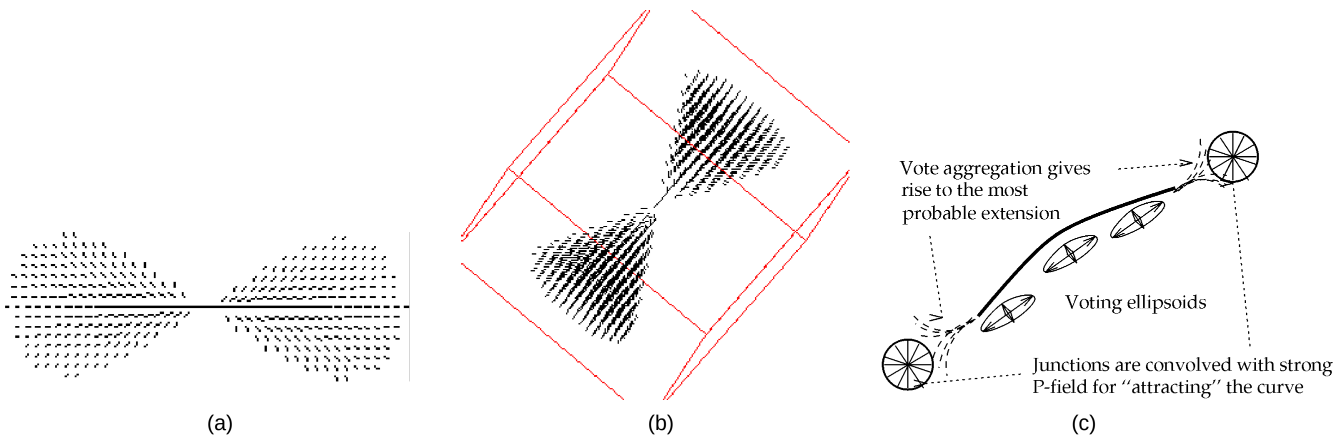

Vote aggregation gives rise to the most probable extension
Voting ellipsoids
Junctions are convolved with strong P-field for "attracting" the curve
(a)
(b)
(c)

Fig. 12. (a) The 2D extension field. (b) Rotating about its "long axis" produces the 3D version. (c) Convolving a curve with the 3D extension field, and junctions with strong P-field, for inferring the most probable extension.

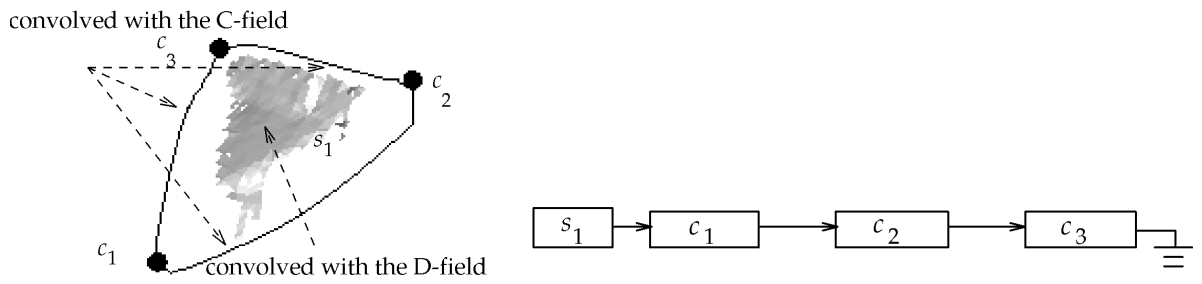


Fig. 13. For surface extension, the corresponding normals from this trimmed surface mesh are convolved with D-field. Curve segments are convolved with C-field. An incidence curve list for the surface.

factor (its choice is not critical; typical factor is two) to the curve inhibitory fields in Section 6.2. Fig. 12c.

Note that we use *two* types of voting fields (namely, the 3D extension field and P-field) in a single voting pass. Since both of them are vector fields, the voting (i.e., vector convolution and aggregation) proceeds in exactly the *same* way as described in Section 3.

For vote interpretation, we assign a two-tuple $(s, \bar{t})$ in each voxel by the following. Since the voting ellipsoid will be very elongated if there is a high agreement in one direction (Fig. 12c), $s = \lambda_{max} - \lambda_{mid}$ is chosen as the saliency, and $\bar{t} = V_{max}$ gives the direction of the curve segment tangent. With this map, a slight modification of the extremal curve extraction algorithm described suffices to extract the desired extended curve. The low threshold is eliminated and curve tracing continues until the junction is exactly hit. This extended curve preserves curve orientation discontinuity.

6.4 Surface Trimming by Inhibitory Curves

In this phase, the extremal surface algorithm is modified to take not only the SMap but also the extended curve obtained in the previous phase to produce a triangular mesh. This is done by:

1) First, the voxels in the SMap corresponding to the location of the extended curve are convolved with a *surface inhibitory field* to protect surface orientation discontinuity detected in the CMap.
2) Then the surface is traced as described in Section 5.3. A set of trimmed surfaces is produced.

6.5 Surface Extension Toward Excitatory Curve

The extended curves and the trimmed surface computed in previous phases are used together to produce an extended surface with preserved surface orientation discontinuity.

First, an incidence curve list is constructed for each trimmed surface. This list corresponds to the set of extended curves with which a trimmed surface will intersect (Fig. 13). Similar to the use of incidence graph, an incidence curve list is used to group relevant surface and curve features so that we can process surface extension in a divide-and-conquer manner and avoid unnecessary interaction among excitatory fields. To create this list for each trimmed surface, we examine each curve, and a curve closest to the mesh boundary of the trimmed surface will be assigned to the list.

Then, for each trimmed surface with its (enclosing) curves, we treat them as input to our voting procedure (i.e., we quantize both the curve (tangents) and the surface (normals). This quantization is needed because both the extended curves and the trimmed surfaces are of subvoxel precision. Two excitatory fields are used in a single voting pass (Fig. 13a).

1) (Excitatory C-field) The tangents are convolved with the C-field, in which vector weight in each vote is increased in order to "attract" the trimmed surface

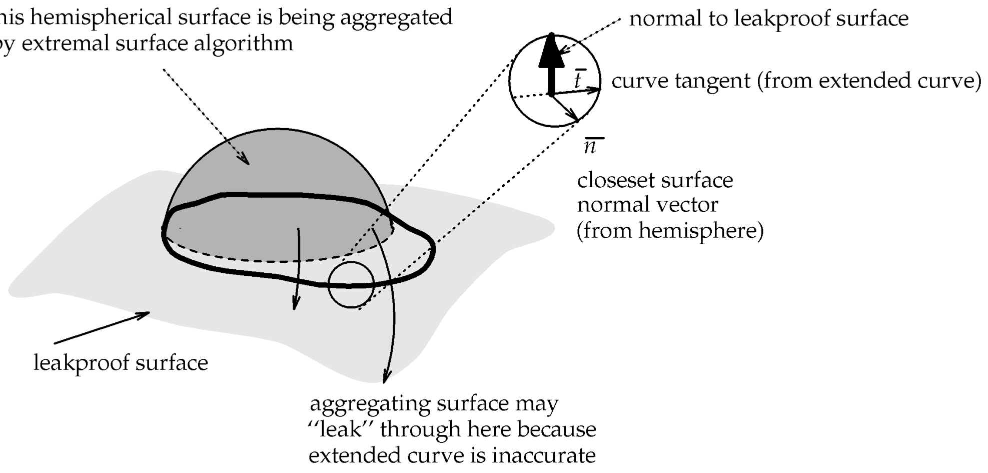

this hemispherical surface is being aggregated
by extremal surface algorithm
normal to leakproof surface
$\bar{t}$
curve tangent (from extended curve)
$\bar{n}$
closeset surface
normal vector
(from hemisphere)
leakproof surface
aggregating surface may
"leak" through here because
extended curve is inaccurate

Fig. 14. A curve lies on a *leakproof surface*, which is obtained by D-field convolution after normal estimation. Leakproof surface is also used later to infer a more accurate curve by surface/surface intersection.

toward the curve. Again, its size is related by the size of the surface inhibitory field used in Section 6.4.

2) (Excitatory D-field) The normals are convolved with the D-field for inferring the most natural extension for filling the gap.

Note that the voting process is exactly the same as described in Section 3, even though we use hybrid vector fields to vote. For vote interpretation, we assign to each voxel a two-tuple $(s, \bar{n})$, where $s = \lambda_{max} - \lambda_{mid}$ is the saliency and $\bar{n} = V_{max}$ gives the direction of normals.

The resultant map, which is a volume of surface normals with their saliencies, is fed into the extremal surface extraction algorithm with the following modification (Fig. 14):

Since the extended curve may be *inaccurate* (recall that no surface information is used at the time when the curve was computed), the evolving surface may extend beyond, or "leak" through, the extended curves if the saliency of the currently visiting voxel in the SMap exceeds the low threshold of the extremal surface algorithm. (Note that, it does not suffice to simply increase the low threshold, nor to inhibit the neighboring voxels around the possibly incorrect curve.)

To prevent such leakage, we infer a *leakproof surface* by using our voting procedure, as in the following:

1) For all tangents $\bar{t}$ of an extended curve, we compute the cross product with its closest surface normal $\bar{n}$ obtained above (Fig. 14).
2) These estimated normals constitute another set of *sparse* data, and the D-field convolution and extremal surface extraction will explicitly produce the leakproof surface. This triangular mesh approximates the surface on which the extended curves are lying, and is used to inhibit the extremal surface algorithm from extraneous extension or "leakage."

At this point, readers may ask why there is no need for leakproof curve or surface in the case of curve extension toward excitatory junction (Section 6.3). First, we do not have enough information for inferring such leakproof curve or surface in that phase. Also, for curve extension, we set a distance threshold to prevent the curve from missing the junction during extension. However, this heuristics may not give the best *intermediate* result; but the *final* result will improve when surface information is taken into account, as in the last phase of the integration to be described in the following.

6.6 Final Curves and Junctions From Surfaces

The extended surfaces obtained in the previous phase are most reliable because they are inferred from cooperating intermediate junction, curve, and surface estimates together. These surfaces are used in turn to generate better intersection curves and junctions which are lying, or localized, on the surfaces. (Recall that intermediate junctions and curves are obtained without taking surfaces into consideration, since the surfaces as detected around those junctions and curves are unreliable.)

By construction, these intermediate curves and their junctions lie on the leakproof surface. Therefore, it suffices to compute the *surface/surface intersection*, or the *precise* surface boundaries, between the extended surface (the most reliable cue) and the leakproof surface (on which curves and junctions are lying). A set of line segments results which, after cleaning up, is the set of coherent curves and junctions where our final surfaces should intersect precisely.

Note that since a curve is usually shared by more than one salient surface, the most salient of all extended surfaces is used to compute the surface/surface intersections with the leakproof surface. And the resulting final curve will be marked so that it will not be computed more than once.

7 Space and Time Complexity Analysis

Let

n = number of input points
k = maximum dimension of voting field
s = number of voxels occupied by output surfaces
c = number of voxels occupied by output curves
j = number of junctions.

In our implementation, the input is quantized but *not* stored in a 3D voxel array. Such a voxel array is very expensive and wasteful if the data set is large and the inferred surface is usually thin. Instead, a heap is used for storing the quantized input. The size of the input heap is $O(n)$.

Since a 3D voxel array is not available, an efficient data structure is needed to maintain the vote collection (i.e., SMap, CMap, and JMap), the output surface patches and curve segments. A red-black tree [2] is used for storing votes. The size is dominated by the SMap, which is $O(s)$ since only voxel occupied by features will be considered by our implementation (vote *gathering* will be described shortly). We also use Fibonacci heaps for storing the seeds for extremal surface and curve extraction. Although their size is also $O(s)$, since they only store the discrete voxel locations and their corresponding surface or curve saliencies, their storage requirement is not as substantial as that of the red-black tree.

Therefore, for space complexity, the heaps and the vote collection together takes $O(s + n)$ space. For sparse data, $n \ll s$. Also, the vote collection stores surface patches of subvoxel precision. Therefore, the storage requirement for the initial heap is not as substantial when compared with that of the red-black tree vote collection. Therefore, the total space requirement is $O(s)$ in practice.

Next, we analyze the time complexity of our method. For input quantization, heap insertion and search, which are all we need, take $O(\log n)$ in time [2]. For the red-black tree operations, insertion, search, and deletion can be done in $O(\log (s + c + j))$ time (or $O(\log s)$ because $s \gg c + j$). Note that deletion is also needed for the maintenance of the vote store, because indiscriminate growth of the tree will lead to severe memory swapping that degrades the computing system. In the current implementation, we limit the maximum size of the vote collection to be 20 MB. When this threshold is exceeded, the whole tree will be purged for freeing the memory. For operations on Fibonacci heaps, insertion and extraction of seed requires $O(1)$ and $O(\log s)$, respectively.

In our implementation, each site *gathers* all the votes cast by its effective neighborhood (size of voting field) only, performs smoothing, computes the eigensystem and surface patch (if any) for that site, all on-the-fly. The result produced by vote casting (as described in Section 3) is equivalent to that produced by vote gathering. Note that when *dense* data is available (during surface extension toward excitatory curves) for which small voting fields can be used, the *effective neighborhood* for each site is also small.

The total time complexity is analyzed as follows:

1) *Computing SMap, CMap, and JMap.* It takes $O(sk^3)$ time for vote convolution, vote aggregation and interpretation, because only voxels occupied by surfaces (and their finite neighborhood that contains curves and junctions) are considered.
2) *Initial junctions from JMap.* $O(s)$ time for local maxima extraction.
3) *Curve trimming by inhibitory junctions.* We preprocess the CMap in $O(s)$ to build a Fibonacci heap (F-heap) for $O(\log s)$-time per seed extraction afterward. For extremal curve algorithm, computing zero crossings using `SingleSubVoxelCMarch` only involves a constant number of choices, so it takes $O(1)$ time to produce a point on an extremal curve (if any). Therefore, the extremal curve algorithm runs in linear time, i.e., at most $O(s + c)$, or $O(s)$ because $s \gg c$ in practice.
4) *Curve extension toward excitatory curves.* It takes $O(c + j)^2$ time to compute the incidence graph. Total vote convolution, aggregation and interpretation takes at most $O((c + j)k^3)$ time.
5) *Surface trimming by inhibitory curves.* Like (3), seed extraction takes $O(\log s)$ time. Also, `SingleSubVoxel-March` takes $O(1)$ time to compute a zero-crossing patch (if any) in a voxel. Therefore, the extremal surface algorithm runs in linear time, i.e., at most $O(s)$.
6) *Surface extension toward excitatory curves.* Incidence curve list can be constructed in $O(s + c)^2$ time. Total vote convolution, aggregation, and interpretation takes at most $O((s + c)k^3)$ time. The extremal surface extraction algorithm runs in time linear with the output size, i.e., at most $O(s)$ time.
7) *Final curves and junctions from surfaces.* The surface/surface intersection routine can be embedded in (6), where each intersection between two voxel surfaces takes $O(1)$ time.

TABLE 1
SPACE AND TIME COMPLEXITIES OF OUR METHOD

Heap (for input quantization)	$O(n)$
Fibonacci heap (for seed store)	$O(s)$
Red-black tree (for vote collection)	$O(s)$
Total space complexity	$O(n + s) \approx O(s)$
Computing SMap, CMap, and JMap	$O(sk^3)$
Initial junctions	$O(s)$
Curve trimming	$O(s + c)$
Curve extension	$O((c + j)k^3)$
Surface trimming	$O(s)$
Surface extension	$O((s + c)k^3)$
Final curves and junctions	$O(s)$
Total time complexity	$\approx O(sk^3)$

In all, the most time consuming part is step 6, because voting is performed on *dense* normals given by SMap, a thick set of points with normal information. Because we have dense information, the voting field used (i.e., k) in step 6 is small (typical size is about $5 \times 5 \times 5$). Our program runs in about 15 minutes on a Sun Sparc Ultra 1 for 1,000 input points.

In summary, we tabulate the space and time complexities in Table 1.

8 RESULTS

We produce synthetic and real sparse input data and present different views of the extracted surface. (The real data are sampled using a 3D digitizer in sweep mode.) Each example emphasizes one or more aspects as described in the following.

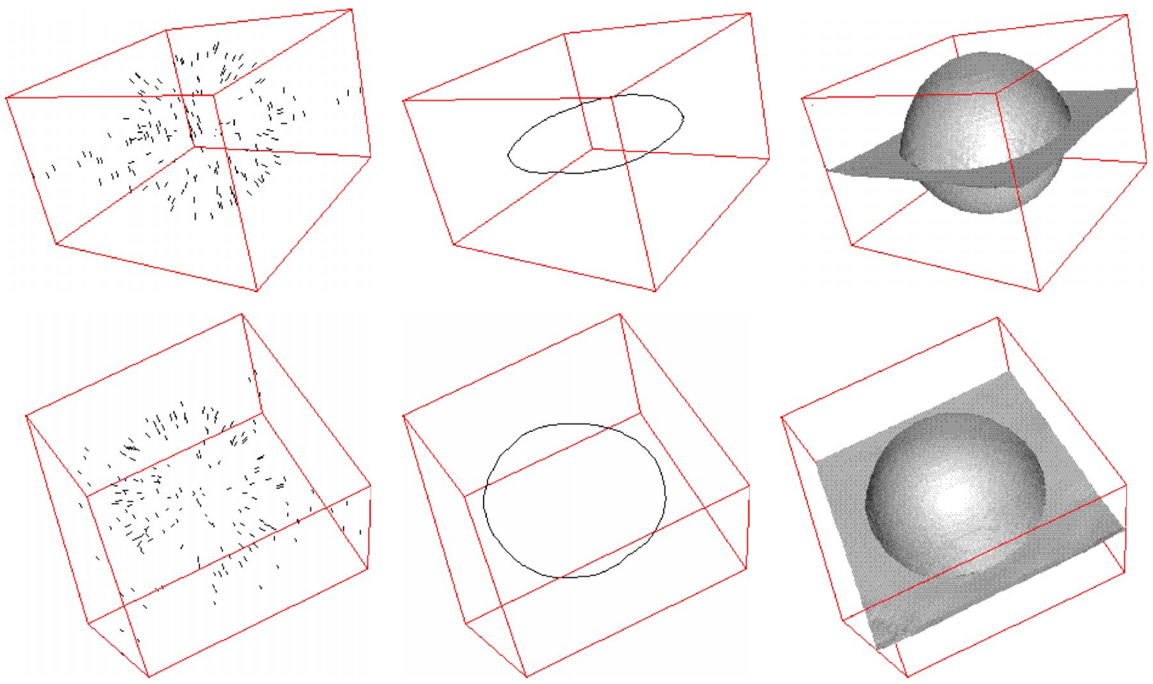


Fig.15. Plane and sphere. An intersection curve is precisely inferred. The integrated surface description consists of an upper and lower hemispherical surfaces, a square planar surface with a circular hole, and a circular planar surface inside the sphere (not visible).

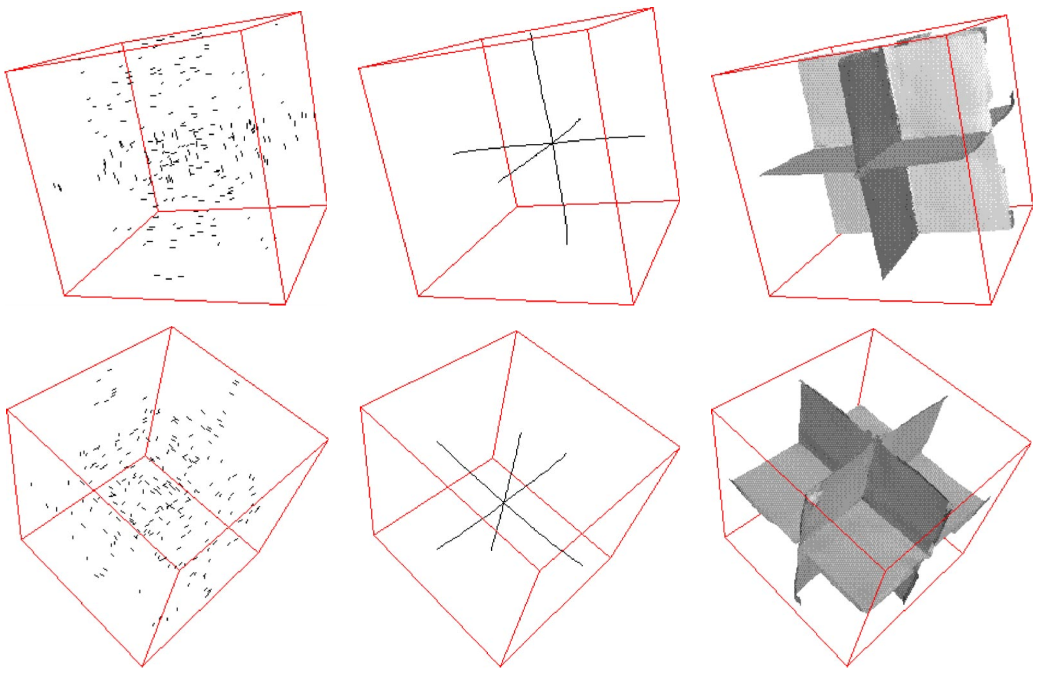


Fig. 16. Three orthogonal planes. There are six extremal curves and one six-junction which are coherently integrated with twelve extremal planar surfaces. The junction curves are properly localized.

8.1 Plane and Sphere

A total of 342 data points with normals are sampled from a sphere intersecting with a plane. Fig. 15 shows two views of the input data, the extracted intersection curve, and integrated result obtained using our cooperative approach.

8.2 Three Planes

A total of 225 data points with normals are sampled from three orthogonal and intersecting planes. Initial estimates are cooperatively refined using our approach. The result is shown in Fig. 16.

8.3 Triangular Wedge

A digitizer is made to sweep over a real triangular wedge and a set of 1,266 data points is produced. Successive digitized points form an edgel, so we use C-field for normal recovery. Then, the cooperative computation is run as described. Result is shown in Fig. 17.

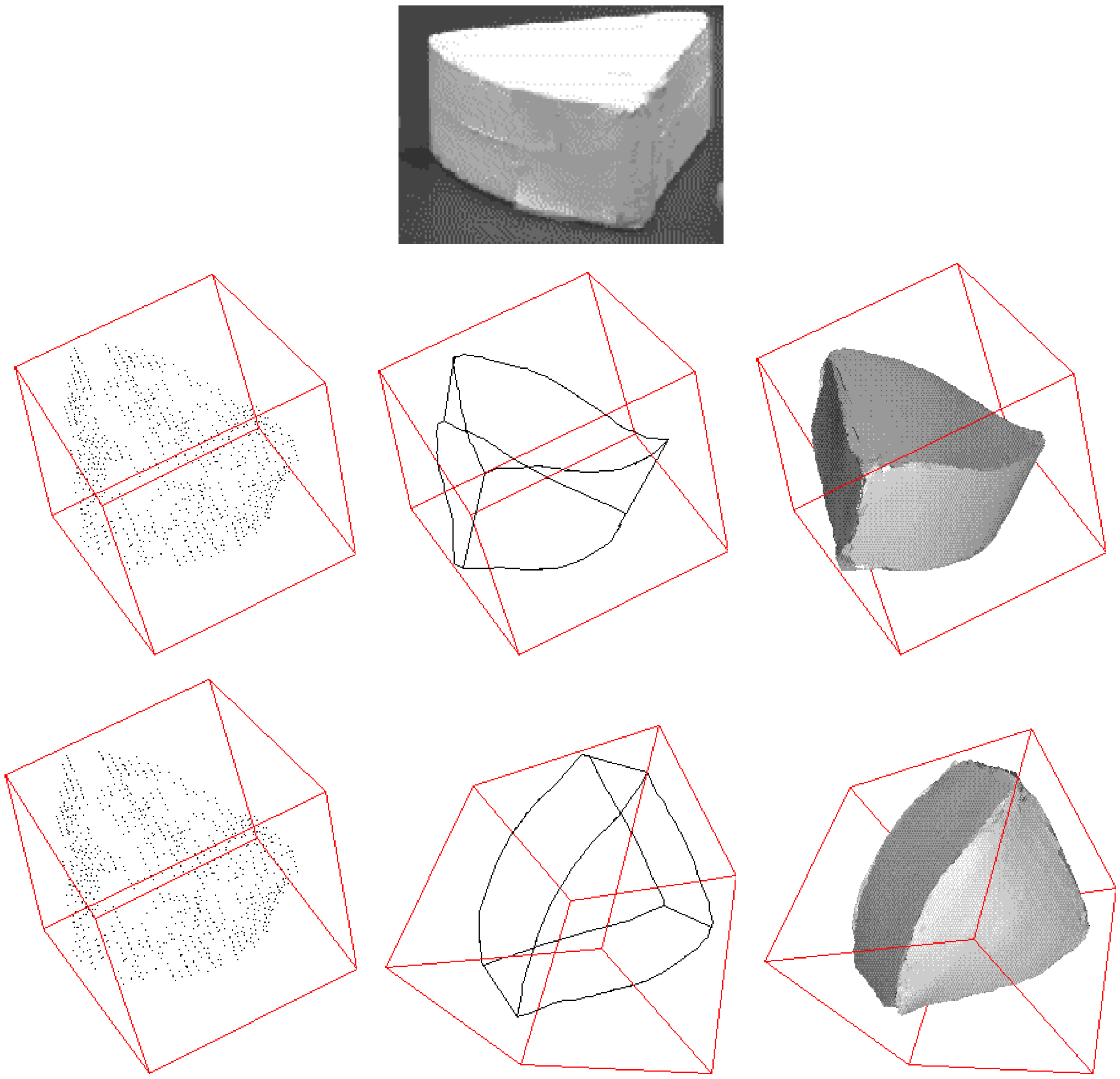


Fig. 17. The surface of a triangular wedge is sampled using a digitizer. The set of points obtained is used as input to our program to generate the integrated descriptions of junctions, curves, and surfaces. Six three-junctions and nine extremal curves, and six extremal surfaces are integrated. The real object is also shown.

8.4 Wooden Block

We again use a digitizer to sample a set of 1473 data points from a wooden block. Successive digitized points form an edgel, so we use C-field for normal recovery. Then, the cooperative computation is run as described. Result is shown in Fig. 18.

8.5 Crown

A set of 24 views of a *Crown* (a dental restoration) is registered using a rotational registration system shown in Fig. 19 (courtesy of the GC Corporation, Japan). This data set contains a total of 60,095 points. We registered all these views using a single coordinate system. P-field is used for normal recovery from the noisy dental imprints, which is followed by D-field convolution and cooperative process as described. The result is shown in Fig. 20. With such rich (but noisy) data, we can detect the upper and lower sides, the creases and the crown, which are in turn used to produce a coherently integrated surface and curve description. This data set is difficult because it consists of two very close-by upper and lower surfaces bounded by the "crown," with many creases which are only implicit in the data. All these are faithfully and explicitly inferred in our 3D description.

Note that we can only integrate the two intersection curves (namely, the preparation line and the fixture that holds the crown) with the crown surfaces. For the crease curves on the upper surface of the crown, since they only converge in a saddle *region*, but not a salient point junction, they are only detected, but not integrated as described.

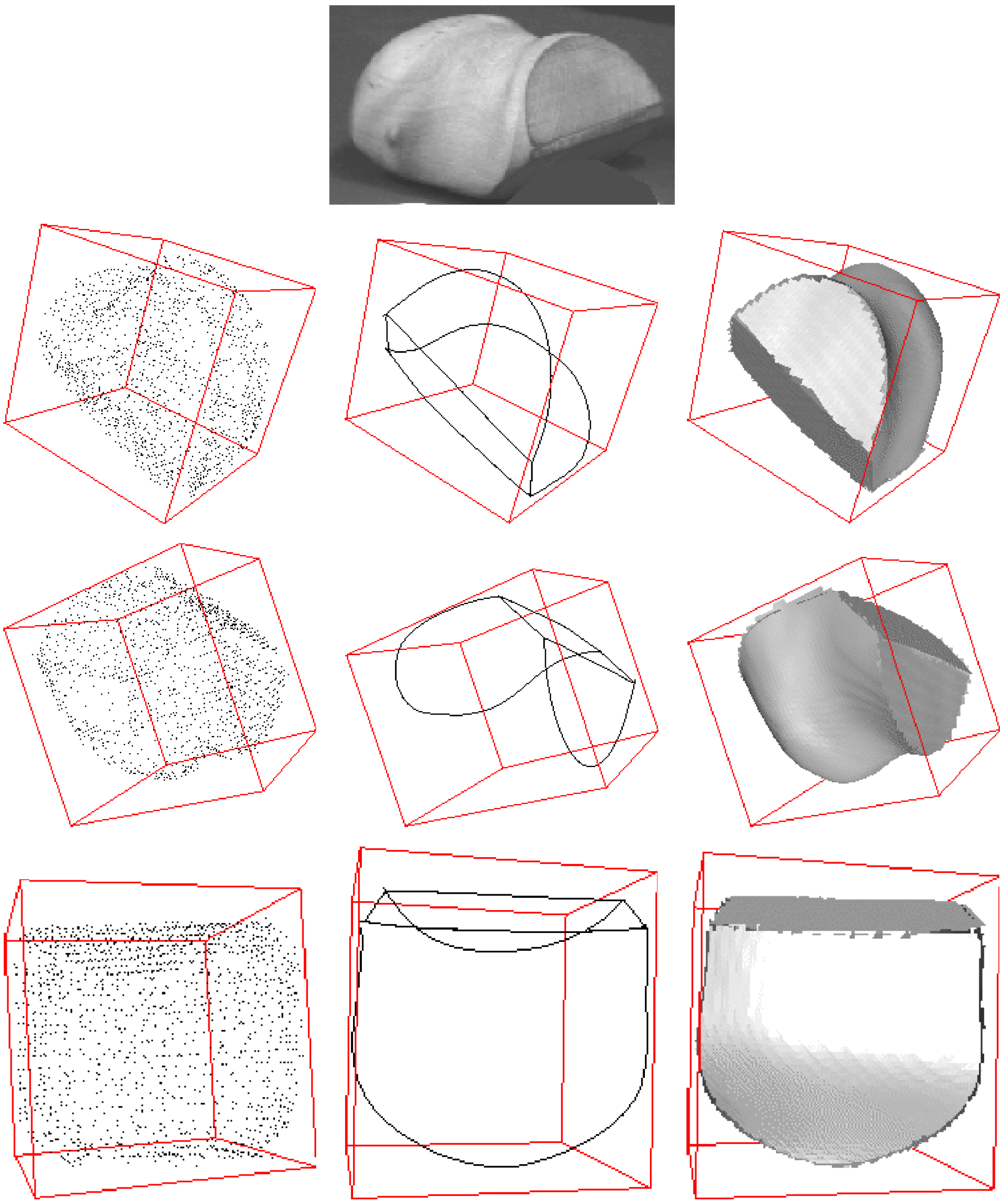


Fig. 18. The surface of a wooden block is sampled using a digitizer. The set of points obtained is used as input to our program to generate the integrated descriptions of junctions, curves, and surfaces. Four three-junctions and six extremal curves are inferred and integrated with four extremal surfaces, some of them have nonconstant curvature. The real object is also shown.

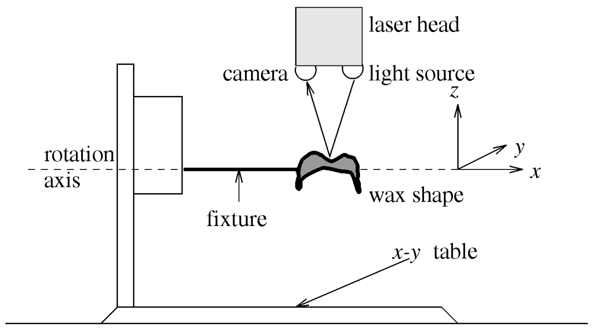


Fig. 19. A rotational registration system (courtesy of the GC Corporation, Japan).

8.6 Book Stack

We test our approach to infer segmented and integrated surface, curve, and junction description from stereo. Details can be found in a recent paper by Lee and Medioni [12]. Fig. 21 depicts the input intensity stereo images and the resultant integrated description. First, we start with an estimate of the 3D disparity array in the traditional manner. Potential edgel correspondences are generated by identifying edge segment pairs that share rasters across the images (Fig. 21a). Initial point and line disparity estimations are then made. To infer salient structure from the disparity array, we perform D-field voting for each matched point, from which the SMap is computed. The most salient match are kept along each line of sight (Fig. 21b), using the unique

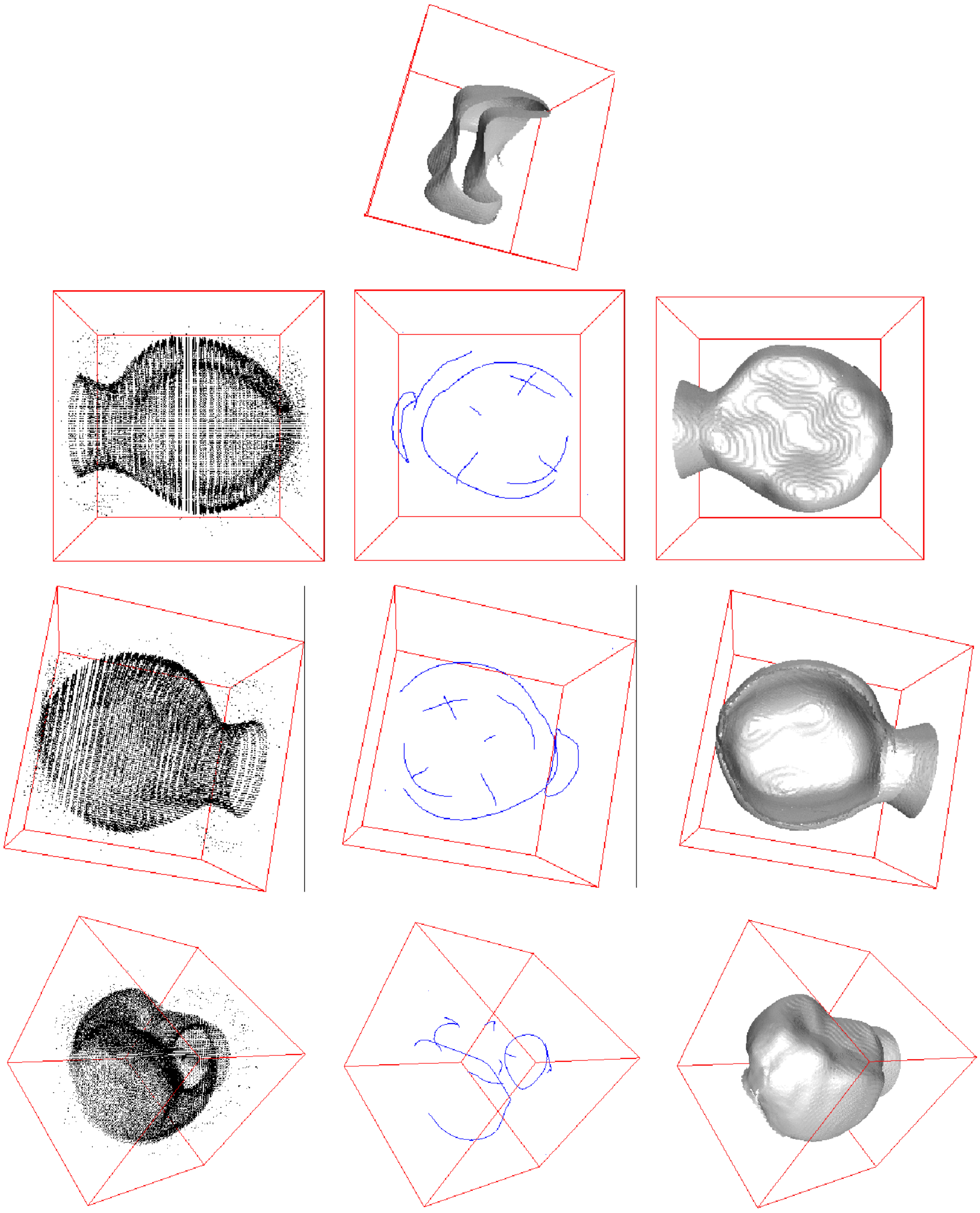
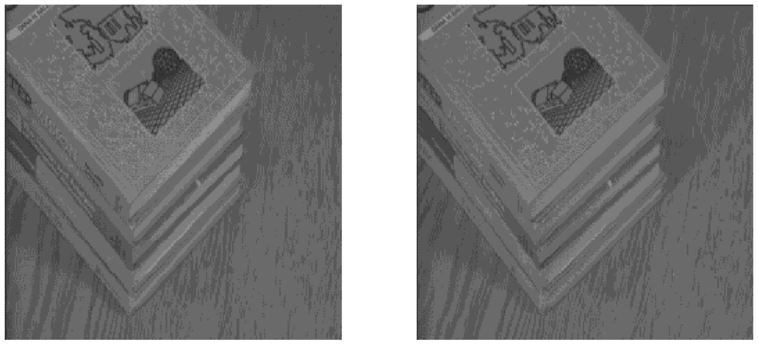


Fig. 20. A crown tooth is sampled using a rotational registration system. The noisy data set is used as input to our program to generate an integrated description consisting of extremal curves and surfaces. A middle slice of the extremal surfaces is also shown at the top.

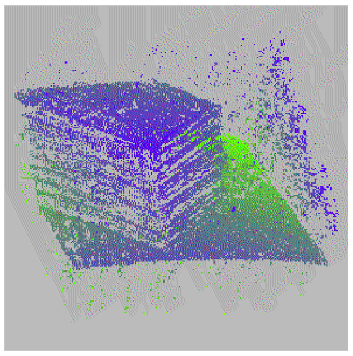
disparity assignment. This filtered, though still noisy, point set is then used as input to our program and the integrated description and the texture mapped surfaces are shown.

9 Conclusion and Future Work

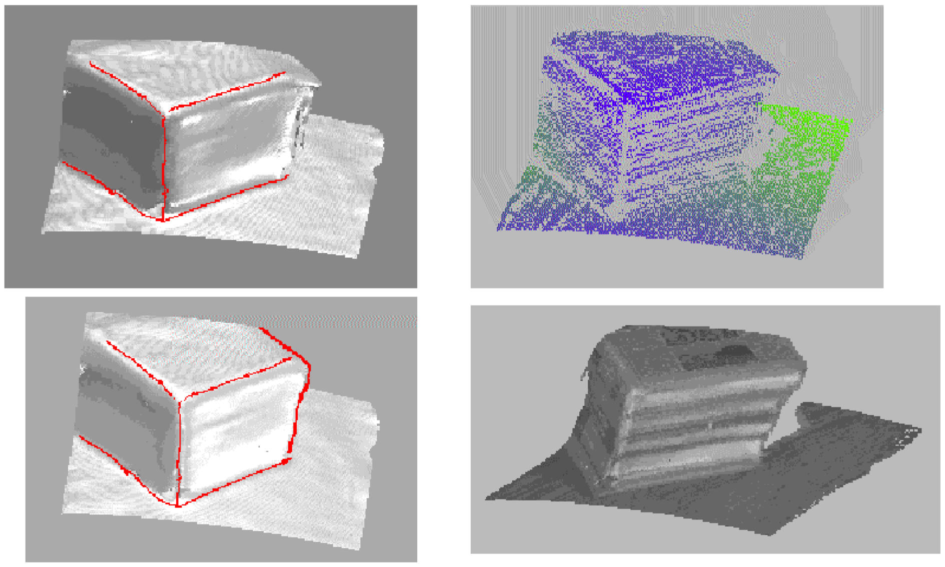
In this paper, we describe a general method for inferring coherent surface, curve, and junction from a sparse 3D data set in which surface and curve orientation discontinuities are explicitly preserved. The cooperative approach refines initial estimates by using excitatory and inhibitory fields (which are derived from an earlier work [6]) and novel surface and curve tracing processes that produce extremal surfaces and curves given dense saliency maps. Although the integration process we described above is complex, it shall produce an integrated surface, curve, and



(a)



(b)



(c)

Fig. 21. Inference of integrated surface, curve, and junction description from stereo. (a) Input stereo images. (b) Noisy point set after assigning unique disparity. (c) Resultant surfaces, curves, and junctions.

junction description if such features are salient enough (not necessarily explicit) in the input. We qualitatively evaluate our method by a variety of convincing examples. Therefore, more work has to be done on the quantitative analysis (e.g., the effect of scale of analysis and amount of additive noise on the reconstruction result) to quantify the stability, existence, and performance bounds of our method.

As indicated in the time complexity analysis, we can reduce the order of the most time-consuming part by voting only at curve endpoints or surface boundary for inferring the natural extension if curvature information can be reliably estimated. Currently, without such information, all normals (resp. tangents) of initial surface (resp. curve) have to vote together in order to infer the most perceptually natural extension. Together with the scale of analysis, which is related to the size of mask, and the extension of the methodology to other problems, are the topics of our current research effort.

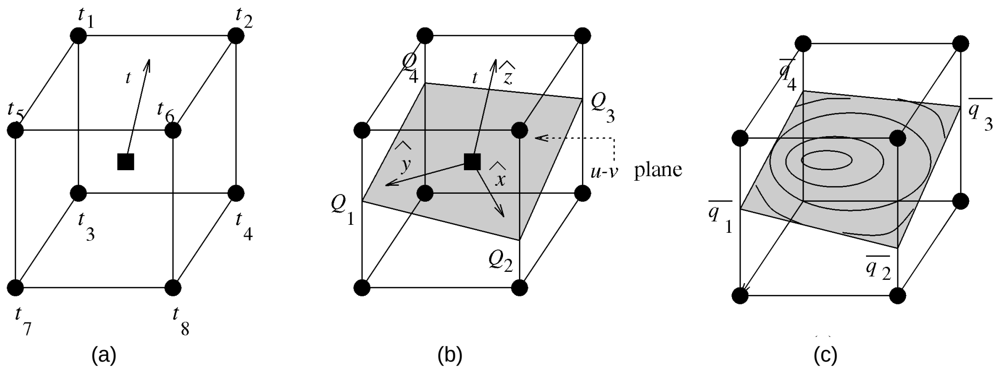


Fig. 22. Illustration of `SingleSubVoxelCMarch`.

APPENDIX

A.1 SingleSubVoxelCMarch

Consider the eight voxels V_r with $(s, \overline{t_r})$, $1 \le r \le 8$, that make up a cuboid. Define the *cuboid tangent*, denoted by $\bar{t}$, to be the interpolated tangent at the center of the cuboid, using the eight $\overline{t_r}$s. We compute the 3D subvoxel coordinates of the point that an extremal curve with tangent $\bar{t}$ will pass through by the following:

Step 1:

1) Translate the unit cuboid to the world origin. Let **T** be the translation matrix.
2) Compute $\bar{g}$ for the eight V_rs, using (5).
3) Compute the cuboid tangent $\bar{t}$ by interpolating the *aligned* $\overline{t_r}$, $1 \le r \le 8$. (Note: Two tangents are aligned if their dot product is nonnegative.) Therefore,

$$\bar{t} = \sum_{r=1}^{8} t_r / 8 \tag{9}$$

(See Fig. 22a). Thus, $\bar{t}$ defines a u-v plane through which an extremal curve with tangent $\bar{t}$ passes. We assume this plane passes through the cuboid center (Fig. 22b).

Step 2:

1) For each cuboid edge (P_0^k, P_1^k), $1 \le k \le 12$, we compute the intersection point (if it exists) with the u-v plane. Solving the corresponding ray-plane equation, an intersection point Q_k on a cuboid edge is given by the parameter u_k (Fig. 22b):

$$u_k = -\frac{\bar{t} \cdot P_0^k}{\bar{t} \cdot (P_1^k - P_0^k)} \tag{10}$$

$$Q_k = P_0^k + (P_1^k - P_0^k) u_k. \tag{11}$$

If $\bar{t} \cdot (P_1^k - P_0^k) = 0$ or $u_k < 0$ or $u_k > 1$, there is no intersection.

2) Order all intersection points $\{Q_k\}$ so that they form a cycle. Since $\{Q_k\}$ lie on the u-v plane, this problem is equivalent to computing a 2D convex hull for $\{Q_k\}$. Several known algorithms are available in any standard algorithms text such as [2]. Let the ordered set be $\{Q_1, \cdots\}$.
3) Define a frame which aligns with u-v plane by a rotation matrix **R**:

$$\mathbf{R} = \begin{bmatrix} \hat{x}^T \\ \hat{y}^T \\ \hat{z}^T \end{bmatrix} \tag{12}$$

with

$$\hat{z} = \bar{t} / \|\bar{t}\| \tag{13}$$

$$\hat{x} = Q_1 / \|Q_1\| \tag{14}$$

$$\hat{y} = \hat{z} \times \hat{x}. \tag{15}$$

We then transform the ordered $\{Q_k\}$ to frame **R**. So for all intersections Q_k, we assign

$$Q_k \leftarrow \mathbf{R} Q_k.$$

See Fig. 22b. Note that after applying **R** to Q_k as above, $(Q_k)_z$ will become zero.

Step 3: For each (ordered) intersection point Q_k which lies on a cuboid edge (P_0^k, P_1^k) connecting two voxels, we compute the $\overline{q_k}$ w.r.t. the frame **R** (Fig. 22c) as follows:

1) Compute the interpolated gradient vector for Q_k, denoted by $\overline{g_k}$. Let the gradient vector at P_0^k and P_1^k be $\overline{g_0^k}$ and $\overline{g_1^k}$, respectively. If $Q_k = P_0^k + (P_1^k - P_0^k) u_k$ is given by (11), then by linear approximation, we have

$$\overline{g_k} = \overline{g_0^k} + \left(\overline{g_1^k} - \overline{g_0^k}\right) u_k. \tag{16}$$

2) Compute $\overline{q_k}$ for each Q_k:

$$\overline{q_k} = \mathbf{R}(\bar{t} \times \overline{g_k}). \tag{17}$$

Step 4: (Marching cycle) Now, $(\overline{q_k})_x$ corresponds to the u component, and $(\overline{q_k})_y$ corresponds to v component of (5), with $(\overline{q_k})_z = 0$. We march along the sides of the cycle in order given by the ordered set $\{Q_k\}$, and compute zero crossings (Fig. 23). Because we approximate a zero crossing by linear interpolation, if there exists an extremal point, the positive and negative $(\overline{q_j})_x$ (resp. $(\overline{q_j})_y$)

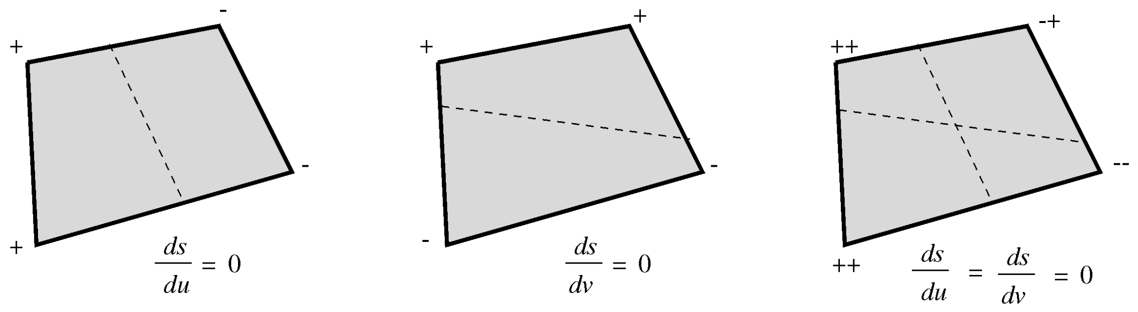


Fig. 23. Linear interpolation gives a subvoxel approximation where $\frac{ds}{du} = \frac{ds}{dv} = 0$.

should be linearly separable and thus four zero crossings, and subsequently two 2D straight lines, will be produced. Their intersection corresponds to the extremal point in frame **R**. Denote this intersection point in frame **R** by $P^{\mathbf{R}}$.

Step 5: Transform $P^{\mathbf{R}}$ back to the world frame **W**, i.e.,

$$P^{\mathbf{W}} = \mathbf{T}^{-1}\,\mathbf{R}^{-1}\,P^{\mathbf{R}}. \tag{18}$$

Both $\mathbf{T}^{-1}$ and $\mathbf{R}^{-1}$ are easy to compute since they are pure translation and rotation matrices, respectively. $P^{\mathbf{W}}$ is the extremal point with subvoxel precision through which an extremal curve will pass.

A.2 SingleSubVoxelMarch

Consider the four voxels which constitute a face of a cuboid. Each voxel is labeled "+" if $q_{i,j,k} \geq 0$ and "–" otherwise. Hence there are $2^4 = 16$ possible configurations which can be reduced to seven by rotational symmetry. Ambiguities are resolved using the method proposed in [15], [23]. We refer readers to these papers for more details. After resolving the ambiguities, the zero crossings will be grouped into cycles and then triangulated.

Acknowledgments

We would like to thank Dr. Mi-Suen Lee for her the result on stereo analysis of the *book stack* in the result section, for which the surfaces, curves, and junctions are extracted using our extremal feature algorithms, and also her new implementation on tensor voting that inspired an improved time and space complexities of the voting procedure originally proposed by Guy and Medioni [6]. This research was supported in part by the U.S. National Science Foundation under Grant No IRI-94231441.

References

[1] J.D. Boissonnat, "Representation of Objects by Triangulating Points in 3-D Space," *Proc. Sixth Int'l Conf. Pattern Recognition*, pp. 830–832, 1982.

[2] T. Cormen, C. Leiserson and R. Rivest, *Introduction to Algorithms*. Cambridge, Mass.: MIT Press.

[3] T.E. Boult and J.R. Kender, "Visual Surface Reconstruction Using Sparse Depth Data," *Proc. Computer Vision and Pattern Recognition* pp. 68–76, Miami Beach, Fla., 1986.

[4] A. Blake and A. Zisserman, "Invariant Surface Reconstruction Using Weak Continuity Constraints," *Proc. Computer Vision and Pattern Recognition*, pp. 62–67, Miami Beach, Fla., 1986.

[5] P. Fua and P. Sander, "Segmenting Unstructured 3D Points Into Surfaces," *Proc. European Conf. Computer Vision*, pp. 676–680, Santa Margherita Ligure, Italy, 1992.

[6] G. Guy and G. Medioni, "Inference of Surfaces, 3D Curves and Junctions from Sparse, Noisy 3D Data," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1,265–1,277, 1997.

[7] S. Han and G. Medioni, "Triangular NURBS Surface Modeling of Scattered Data," *Proc. IEEE Visualization '96*, pp. 295–302, 1996.

[8] A. Hilton, J. Stoddart, J. Illingworth, and T. Windeatt, "Implicit Surface-Based Geometric Fusion," *Computer Vision and Image Understanding*, vol. 69, no. 3, pp. 273–291, 1998.

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction From Unorganized Points," *Computer Graphics (SIGGRAPH '92 Proc.)*, pp. 71–78, 1992.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Mesh Optimization," *Computer Graphics (SIGGRAPH '93 Proc.)*, pp. 19–26, 1993.

[11] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int'l J. Computer Vision*, pp. 321–331, 1988.

[12] M.S. Lee and G. Medioni, "Inferring Segmented Surface Description from Stereo Data," *Proc. Computer Vision and Pattern Recognition*, pp. 346–352, Santa Barbara, Calif., June 1998.

[13] C. Liao and G. Medioni, "Surface Approximation of a Cloud of 3D Points," *CAD94 Workshop*, Pittsburgh, Penn., 1994.

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

[15] G.M. Nielson and B. Hamann, "The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes," *Proc. IEEE Visualization '91*, pp. 83–90, 1991.

[16] T. Poggio and F. Girosi, "A Theory of Networks for Learning," *Science*, pp. 978–982, 1990.

[17] G. Roth and E. Wibowo, "An Efficient Volumetric Method for Building Closed Triangular Meshes," *Graphics Interface '97*, pp. 173–180, Kelowna, BC, Canada.

[18] R. Szeliski, D. Tonnesen, and D. Terzopoulos, "Modeling Surfaces of Arbitrary Topology with Dynamic Particles," *Proc. IEEE Computer Vision and Pattern Recognition*, pp. 82–85, New York, NY, June 1993.

[19] C.-K. Tang and G. Medioni, "Integrated Surface, Curve, and Junction Inference for Sparse 3D Data Sets," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 818–824, Bombay, India, Jan. 1998.

[20] D. Terzopoulos and D. Metaxas, "Dynamic 3D Models With Local and Global Deformations: Deformable Superquadrics," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 91–123, July 1991.

[21] D. Terzopoulos and M. Vasilescu, "Sampling and Reconstruction With Adaptive Meshes," *Proc. IEEE Computer Vision and Pattern Recognition*, pp. 70–75, Lahaina, Maui, Haw., June 1991.

[22] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion," *Artificial Intelligence*, vol. 36, pp. 91–123, 1988.

[23] J.P. Thirion and A. Gourdon, "The 3D Marching Lines Algorithm," *Graphic Models and Image Processing*, vol. 58, no. 6, pp. 503–509, 1996.

[24] M. Vasilescu and D. Terzopoulos, "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Hierarchical Subdivision," *Proc. Computer Vision and Pattern Recognition*, pp. 829–832, 1992.

[25] M. Wheeler, Y. Sato and K. Ikeuchi, "Consensus Surfaces for Modeling 3D Objects from Multiple Range Images," *Proc. IEEE Int'l Conf. Computer Vision*, pp. 917–924, Bombay, India, Jan. 1998.

Chi-Keung Tang is currently a PhD candidate at the Computer Science Department, University of Southern California (USC), working under the guidance of Dr. Gérard Medioni in the areas of computer vision and scientific visualization. Prior to his study at USC in 1995, he studied parallel realistic image synthesis techniques in computer graphics, and geometric optimization algorithms in computational geometry. He received his BSc (Hons.) degree from the Chinese University of Hong Kong (CUHK) in 1992, and his MPhil degree from the Hong Kong University of Science and Technology (HKUST) in 1994, all in computer science. From 1993 to 1995, he was a demonstrator at HKUST.

Gérard Medioni received the Diplôme d'Ingénieur Civil from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 1977, and the MS and PhD degrees in computer science from the University of Southern California, Los Angeles, in 1980 and 1983, respectively. He has been with the University of Southern California (USC) in Los Angeles, since 1983, where he is currently associate professor of computer science and electrical engineering. His research interests cover a broad spectrum of the computer vision field, and he has studied techniques for edge detection, perceptual grouping, shape description, stereo analysis, range image understanding, image to map correspondence, object recognition, and image sequence analysis. He has published more than 100 papers in conference proceedings and journals.

Dr. Medioni is a senior member of the IEEE. He has served on program committees of many major vision conferences, and was program cochairman of the *IEEE Computer Vision and Pattern Recognition Conference* in 1991, program cochairman of the *IEEE Symposium on Computer Vision* held in Coral Gables, Florida, in November 1995, general cochair of the *IEEE Computer Vision and Pattern Recognition* Conference in 1997 in Puerto Rico, and program cochair of the *International Conference on Pattern Recognition* held in Brisbane, Australia, in August 1998. Dr. Medioni is associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *Pattern Recognition and Image Analysis* journals. He is also one of the North American editors for the *Image and Vision Computing* journal.