Integrated Surface, Curve and Junction Inference from Sparse 3-D Data Sets

Chi-Keung Tang and Gérard Medioni
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California 90089-0273.
Email: {chitang,medioni}@iris.usc.edu.

Abstract

We are interested in descriptions of 3-D data sets, as obtained from stereo or a 3-D 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, curves, and junctions from a sparse point set. While the method described in [5], [6] provides excellent results for smooth structures, it only detects discontinuities but does not localize them. For precise localization, we propose a non-iterative cooperative algorithm in which surfaces, curves, and junctions work together: Initial estimates are computed based on [5], [6], 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 maximal surface and curve marching algorithms for initial surface and curve extraction. Refinement of initial estimates is achieved by hybrid voting using excitatory and inhibitory fields for inferring reliable and natural extension so that surface/curve and curve/junction discontinuities are preserved. Results on several synthetic as well as real data sets are presented.

1 Introduction

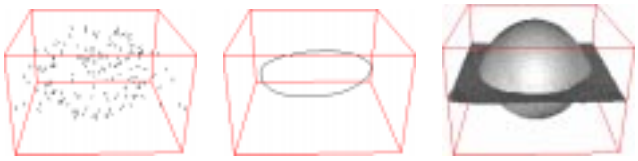

Fig. 1. Inferring integrated high-level description. (a) Input sparse data points with normals. (b) Surface discontinuity is localized as an intersection curve. (c) Output surface with discontinuity preserved.

Our human visual system can perform an amazingly good job of perceiving surfaces from a set of 3-D points. We not only can infer surfaces, but also segment the scene, detect surface orientation and discontinuities. For example, Figure 1 shows a sparse set of points (with normals) sampled from a planar surface intersecting with a sphere. The circle represents the intersection contour which 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 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 [5], [6] proposed to *detect* the presence of junctions and intersection curves from such data. While it did a good job of it, 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 [5], [6], where a non-linear voting process (implemented as a convolution of input features with a pre-defined 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 [5], [6] on this problem, then motivate and describe our approach, and finally show results on complex data.

2 Previous work

Much work has been done in surface fitting to clouds of points. The majority of works use the *deformable model* approach (first proposed by Kass in *et al.* in [9] and [18]) which attempts to deform an initial shape using energy minimization so that the deformed shape fits a set of points. Boult and Kender [2] addressed the sparse data problem specifically and demonstrated a method using minimization over Hilbert spaces. Poggio and Girosi [13] formulated the single-surface approximation problem as a network learning problem. Blake and Zisserman [3] addressed similar problems dealing explicitly with discontinuities. Fua and Sander [4] proposed an algorithm to describe surfaces from a set of points using local properties. Szeliski *et al.* [14] make use of a physically-based dynamic local process evolved from each data point and subject to various forces to deal with a *single* object of unrestricted topology. In [7], [11], 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 [16], [17], and [20]) 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.* [8] 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.

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 discontinuities are usually smoothed out; does not work in the presence of noise. Guy

This work was supported in part by the National Science Foundation under Grant No IRI-94231441.

and Medioni [5] have recently proposed a method to attack these problems. Since our work is a direct extension, we describe it in more detail to serve as background, as well as a source of terminology that will be used throughout this paper. Our novel surface and curve extraction algorithms are also introduced here.

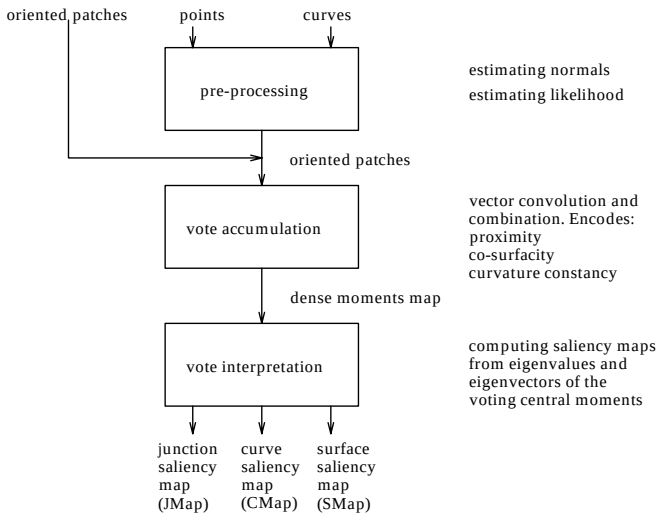

Fig. 2. Flow chart of the method proposed by Guy and Medioni.

Figure 2 shows the major steps of the algorithm. Each input site in the sparse data set is quantized in a 3-D array. A surface normal or tangent may 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*.

3.1 Fields

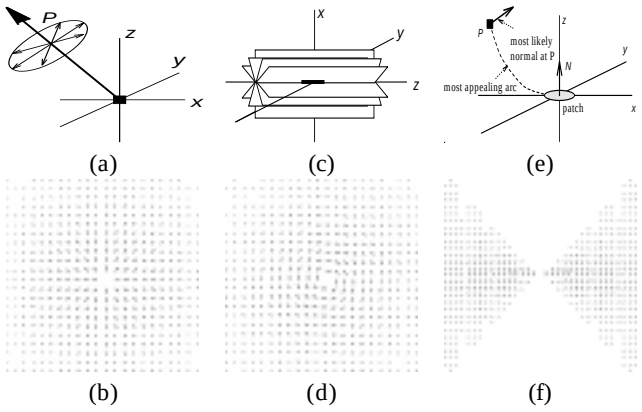

Fig. 3. (a) Given a point P, all normals (thin arrows) at point P are equally likely. We choose to represent all of them with one vector (thick arrow). (b) Cross section of P-field at $y = 0$. (c) Given a curve segment, all planes containing this segment are equally likely. (d) Cross section of C-field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

There are three types of 3-D vector fields proposed in [5], [6]: the *3-D point field* (*P-field*), *curve segment field* (*C-field*), and *Diabolo field* (*D-field*). The first two are used in preprocessing for normal estimation and the third one was used for surface inference.

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 $\overline{OP}$ (Figure 3(a)).

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

D-field. Given a normal vector N at origin O and a point P (Figure 3(e)), without any *a priori* information, the most likely and "natural" 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* (Figure 3(f)).

In all cases, the weight of each vector is inversely proportional to the distance from O and the curvature of the underlying circular arc connecting O and P. A Gaussian decay function is used for that purpose.

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*. Computing saliency maps is done by voting, which is realized by *convolving* each input normal vector with 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, which is equivalent to computing a 3-D 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 site in our 3-D array we decompose the 3x3 variance-covariance matrix of central moments into the corresponding eigensystem, as expressed in Equation (1), where λ_{max}, λ_{mid}, and λ_{min} represent the three sorted eigenvalues of the system, and the V's denote the corresponding eigenvectors of the system. Such decomposition will always yield real, non-negative eigenvalues since the matrix is positive semi-definite:

$$\begin{bmatrix} V_{min} \\ V_{mid} \\ V_{max} \end{bmatrix}^T \begin{bmatrix} \lambda_{min} & 0 & 0 \\ 0 & \lambda_{mid} & 0 \\ 0 & 0 & \lambda_{max} \end{bmatrix} \begin{bmatrix} V_{min} \\ V_{mid} \\ V_{max} \end{bmatrix} \quad (1)$$

The three eigenvectors correspond to the three principal directions of an ellipsoid in 3-D, while the eigenvalues describe the strength and agreement measures of the 3-D votes. On a smooth surface, the votes produce high agreement around one direction, so $\lambda_{max} \gg \lambda_{mid}, \lambda_{min}$ (Figure 4(a)). Along the curve bounding two surfaces, two of the eigenvalues are high, and one small, leading to $\lambda_{mid} \gg \lambda_{min}$ (Figure 4(b)). Finally, at a junction of two or more curves, all three values are similar (Figure 4(c)). Thus three voxel maps defining the surface, curve, and junction *saliencies*, respectively are proposed. Each voxel of these maps has a 2-tuple $(s, \overline{v})$, where s is a scalar and $\overline{v}$ is a unit vector:

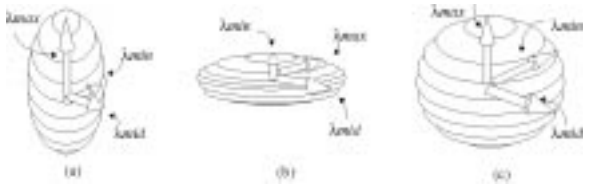

Fig. 4. The three important voting ellipsoids.

- 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.

Pre-processing for the non-oriented 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}.

3.3 High level feature extraction

We now have three 2-tuple $(s, \overline{v})$ per input site. We outline our novel maximal surface and curve extraction algorithms to extract the salient features which correspond to local maxima of the three saliency maps. Details can be found in [15].

3.3.1 *Maximal junction extraction*

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

3.3.2 *Maximal curve extraction*

Each voxel in the CMap holds a 2-tuple $(s, \overline{v})$, where the (scalar) saliency measure s is $\lambda_{mid} - \lambda_{min}$ and $\overline{v} = V_{min}$. We will denote $\overline{v}$ by $\overline{t}$ because $\overline{v}$ is a tangent. Consider the continuous version of the problem of extracting salient curve from a CMap, in which $(s, \overline{t})$ is defined for every point p in 3-D space. A point p with $(s, \overline{t})$ is on a *maximal curve* if any displacement from p on the plane normal to $\overline{t}$ will result in a lower s value. So, a point p lies on a maximal curve if at point p the differential property $\frac{ds}{du} = \frac{ds}{dv} = 0$ holds, where u and v define the plane normal to $\overline{t}$. This definition therefore involves the detection of zero-crossing in the *u-v* plane normal to $\overline{t}$. To do this, we introduce the gradient vector $\overline{g}$ computed on the strengths of adjacent saliencies,

$$\overline{g} = \left[\begin{array}{ccc} \frac{ds}{dx} & \frac{ds}{dy} & \frac{ds}{dz} \end{array} \right]^T \qquad (2)$$

and define $\overline{q} = (\overline{t} \times \overline{g}) \times \overline{t}$. By construction, $\overline{q}$ is the projection of $\overline{g}$ onto the plane normal to $\overline{t}$. Therefore, a maximal curve is the locus of points for which $\overline{q} = \overline{0}$. We can define the corresponding discrete version of $\overline{q}$, and for all eight vertices of a voxel we compute $\overline{q}$'s. The signs of the elements in $\overline{q}$ indicate whether there is any zero-crossing (and thus curve segment) passing through that voxel.

So the maximal curve algorithm picks the seed voxel with $(s, \overline{t})$ whose s value is largest so far, computes a curve segment (if it exists) by using $\overline{q}$, and aggregates the curve in direction $\overline{t}$ until the current s value falls below a low threshold. Denote this curve thus obtained by C_1. Then the algorithm goes back to the seed and repeat 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 maximal curve.

3.3.3 *Maximal surface extraction*

Each voxel in the SMap holds a 2-tuple $(s, \overline{v})$ where $s = \lambda_{max} - \lambda_{mid}$ and $\overline{v}$ is the unit vector V_{max}. We denote $\overline{v}$ by $\overline{n}$ since it is a normal. As before, we consider the continuous version of the problem of salient surface extraction, in which $(s, \overline{n})$ is defined for every point p in 3-D space. A point is on a *maximal surface* if its saliency s is locally maximal along the direction of the normal. That is, a point p lies on a maximal surface if at point p the differential property $\frac{ds}{d\overline{n}} = 0$ holds. This definition involves the detection of zero-crossing on the line aligned with $\overline{n}$, which is computed by defining a scalar $q = \overline{n} \cdot \overline{g}$, where $\overline{g}$ was defined earlier by Equation (2). Therefore, a maximal surface is the locus of points for which $q = 0$. As before, we can define the corresponding discrete version for q, which can be processed directly by the Marching Cubes algorithm [12]: For all eight vertices of a voxel we compute q, whose signs indicate whether there is any zero-crossing (and thus a surface) present in that voxel.

So the algorithm picks the voxel whose s value is largest so far, computes a surface (if it exists) by using q, and aggregates the surface in its neighborhood until the current s value falls below a low threshold. A polygonal mesh is thus produced.

4 Motivation and overall strategy

An integrated high level description requires that discontinuities be explicitly preserved. However, as readily seen from [5], [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 either be smoothed out, left "undecided", or even incorrect in their independent surface and curve descriptions. In the first case, for example, although a

(a) (b) (c)

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

salient curve may be *detected* in CMap, it is still very 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, as shown in Figure 5. In the second case, voxels around discontinuities have a low surface saliency and thus no surface will be produced, creating a gap (Figure 6(a)). Furthermore, because of data sparsity, using the CMap *alone* to trace a curve

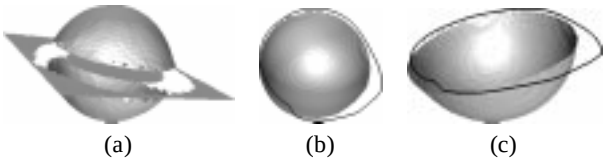

Fig. 6. Incorrect curve may be obtained by considering the CMap alone, (a) gaps may be produced around region of low surface saliency, (b) and (c) two views of the incorrect curve owing to data sparsity.

may produce incorrect result. A sphere intersects 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 Figure 6 where one of the hemispherical surface is also shown).

We have illustrated through examples the limitations of noncooperative feature inference. We here show that while the original work by Guy and Medioni [5], [6] does not handle an integrated description, their approach (viz vector convolution, vote aggregation and interpretation) can be extended cleanly into a *cooperative* framework such that surfaces, curves, and junctions can be inferred cooperatively (and accurately). 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 discontinuities.
- Making use of our novel maximal surface / curve extraction outlined briefly above for initial estimate generation and subsequent refinement.

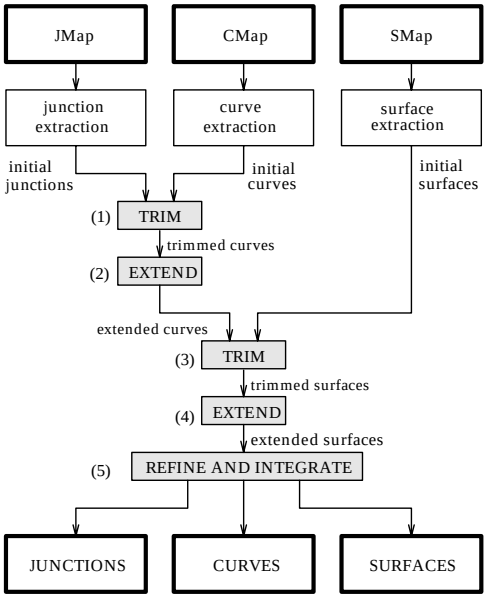

Fig. 7. Overview of cooperative algorithm.

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

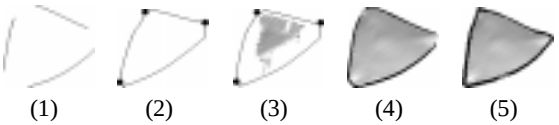

Fig. 8. Illustration of overall strategy.

1. *Curve refinement from inhibitory junctions and CMap.* 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. (Figure 8-(1)).
2. *Curve refinement from excitatory curve and junctions.* Initial junctions and curve are convolved with *curve excitatory fields* so that the curve obtained in (1) are brought to intersect with the junctions (Figure 8-(2)).
3. *Surface refinement from inhibitory curve and SMap.* Each curve segment obtained in (2) is 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. (Figure 8-(3)).
4. *Surface refinement from excitatory curves and surface.* The refined curve and the initial surface are convolved with *surface excitatory fields* such that the surface obtained in (3) can be naturally extended to hit the curve (Figure 8-(4)).
5. *Final curves and junctions from refined surfaces.* The set of surface boundaries obtained by refined surface intersection produces a set of refined curves and junctions which are coherently integrated and localized. (Figure 8-(5)).

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, 3-D curves, and surfaces may change considerably throughout the cooperation process.

5 Cooperative computations using voting

We describe the issues pertinent to cooperative computation, which makes use of hybrid voting and relies on initial surface and curve estimates produced by maximal feature extraction algorithms described in earlier sections.

5.1 Curve refinement from inhibitory junctions and CMap

The maximal curve extraction algorithm outlined in section 3.3.2 is slightly modified such that, besides taking a CMap, it also takes the initial junction estimates as input. It proceeds as follow: Firstly, the tangents of the voxels in CMap in the vicinity of initial junctions are convolved with a *curve inhibitory field*, which inhibits the neighborhood voxels to avoid smoothing out curve discontinuity when a developing curve is evolving. After this stage, a subvoxel linear approximation of the initial curve is produced.

5.2 Curve refinement from excitatory curve and junctions

We make use of the initial curve and junctions to produce a refined curve such that curve discontinuity is preserved. First, by proximity, an incidence graph $G = (V, E)$ is constructed with E correspond to curve and V to incident junctions (Figure 9).

Then, for each curve (edge in E) and its incident junctions (vertices in V), we quantize them as input. Two excitatory

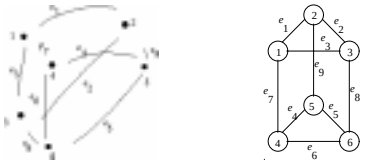

Fig. 9. Given initial curves and junctions, an incidence graph is constructed.

fields are used to refine the curve such that it will intersect the junction precisely in a single voting pass:

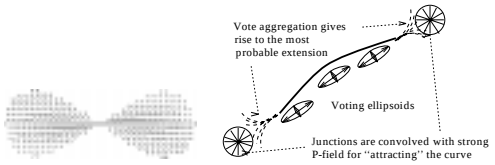

Fig. 10. (a) Projection of the 3-D extension field. (b) Convolving a curve with the 3-D extension field, and junctions with strong P-field, for inferring the most probable extension.

1. Each curve segment (in E) is convolved with the *3-D extension field*, which is a 2-D version of the dual D-field rotated about the x-axis (Figure 10(a)).
2. The junctions (in V) are convolved with the P-field, where each vote in P-field is made heavier (i.e. excitatory) by multiplying each vector weight in the P-field by five in order to "attract" the curve toward the junction. This multiplying factor is determined by the size of the inhibitory field. (Figure 10(b)).

Given the above fields, the voting (vector convolution and aggregation) proceeds in exactly the *same* way as described in section 3, despite that we use *two* types of fields; each in fact is a collection of non-distinguishable vectors used to compute the moments and then diagonalization in Equation (1).

For vote interpretation, we assign the 2-tuple in each voxel $(s, \overline{t})$ as follow. Since the voting ellipsoid will be very elongated if there is a high agreement in one direction (Figure 10(b)), $s = \lambda_{max} - \lambda_{min}$ is chosen as the saliency, and $\overline{t} = V_{max}$ gives the direction of the curve segment tangent. With this map, a slight modification of the maximal curve extraction algorithm described suffices to extract the desired refined curve, which in fact eliminates the low threshold and continues tracing the curve until the junction is exactly hit. This intermediate refined curve thus preserves curve discontinuity.

5.3 Surface refinement from inhibitory curves and SMap

The maximal surface algorithm is modified to take not only the SMap but also the refined curve obtained above to produce a triangulated mesh. Firstly, the locations in the SMap in the vicinity of the refined curve are convolved with a *surface inhibitory field* which, similar to the curve inhibitory field, inhibits the voxels around the detected surface discontinuity. Then the surface is traced as outlined in section 3.3.3.

5.4 Surface refinement from excitatory curve and surfaces

The refined curves and the initial surface computed previously will be used together to produce a refined surface with preserved surface discontinuity. Firstly, by proximity, an incidence curve list is constructed for *each* initial surface; this list points at the set of curves that will be intersected by the corresponding surface (Figure 11).

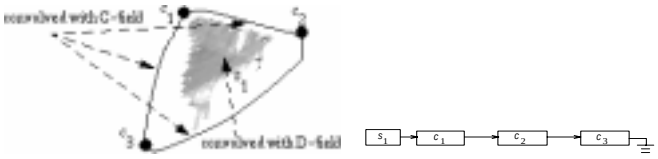

Fig. 11. For surface-to-curve extension, the corresponding normals from this surface are convolved with D-field. Curve segment are convolved with C-field. An incidence curve list for the surface.

Then, for each initial surface with its (enclosing) curves, we treat them as input (i.e., we quantize both the curve (tangents) and the surface (normals).) Two excitatory fields are used to refine the initial surface in a single voting pass (Figure 11(a)).

1. The tangents are convolved with the C-field, in which each vote in the C-field is multiplied by five in order to "attract" the surface toward the curve segments.
2. The normals are convolved with the D-field for inferring the most natural extension for filling the gap.

Vote aggregation and convolution are exactly the same as described in section 3, even though we use hybrid fields to vote. For vote interpretation, we assign to each voxel a 2-tuple $(s, \overline{n})$, where $s = \lambda_{max} - \lambda_{min}$ is the saliency and $\overline{n} = V_{max}$ gives the direction of normals. The total volume of surface normals $\overline{n}$ thus obtained are used as input to the maximal surface extraction algorithm with the following modification (Figure 12): Since the refined curve may be *inaccurate* (recall that no *explicit* surface information is available at the time when the refined curve was computed), the evolving surface may extend beyond, or "leak" through, the curves if the saliency of the currently visiting voxel exceeds the low threshold designated by the maximal curve algorithm.

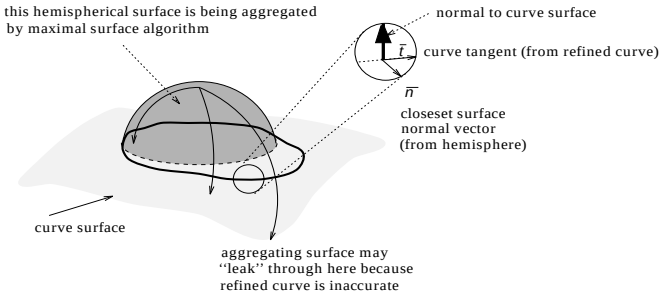

Fig. 12. A curve lies on a *curve surface*, which is obtained by D-field convolution after normal estimation. Curve surface is also used later to infer a more accurate curve by surface / surface intersection.

We infer a "leak-proof" *curve surface* on which the refined curve is lying as follow: For all tangents $\overline{t}$ of the refined curve segment, we compute the cross product with its closest surface normal $\overline{n}$ obtained above (Figure 12). These estimated normals constitute another set of *sparse* data, and the D-field convolution and maximal surface extraction will explicitly produce the curve surface. This surface is used to block the maximal surface algorithm from extraneous extension or "leaking".

5.5 Final curves and junctions from surface

The *refined* surfaces obtained in the previous stage are most reliable because they are inferred from cooperating *intermediate* junction, curve and surface estimates together. However, coherent integration has not yet achieved because such intermediate junctions and curves are obtained without taking surfaces into consideration as they are still unavailable at the time they are computed. By construction, these intermediate curve and its junctions lie *on* the curve surface (introduced in last section). Therefore, it suffices to compute the *surface/surface intersection*, or the *precise* surface boundaries, between the refined surface (the most reliable cue) and curve surface (on which curves and junctions are lying). A set of line segments results which, after cleaning up, are 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 refined surfaces is used to compute the surface / surface intersections with the curve surface. And the resulting final curve will be marked so that it will not be computed more than once.

6 Time complexity analysis

We analyze the time complexity in this section. Let

n = number of input points
d = maximum dimension of voxel array
k = maximum dimension of convolution field
s = number of output surfaces
c = number of output curves
j = number of output junctions
t = maximum output size (in voxels)

The total time complexity is the sum of the following:

1. *Computing SMap, CMap, and JMap.* It takes $O(nk^3)$ time for vote convolution, and $O(d^3)$ time for vote aggregation and interpretation. (Every voxel must be considered.) Thus the total time is $O(nk^3 + d^3)$.
2. *Initial junctions from JMap.* $O(d^3)$ time for local maxima extraction.
3. *Initial curve from inhibitory junctions and CMap.* $O(d^3)$ preprocessing time is spent on building a Fibonacci heap for $O(1)$-time seed extraction. Operations for zero-crossing detection in each candidate voxel take $O(1)$ time. Therefore, the maximal curve extraction algorithm runs in time linear with the output size, i.e. at most $O(ct)$.
4. *Refined curve from excitatory curve and junctions.* It takes $O(c + j)^2$ time to compute the incidence graph. Total vote convolution, aggregation and interpretation takes at most $O((c + j)tk^3 + d^3)$ time.
5. *Initial surface from inhibitory curve and SMap.* Like (3), seed extraction and operation on a voxel to find zero-crossings takes $O(1)$ time respectively. The maximal surface extraction algorithm runs in time linear with the output size, i.e. at most $O(st)$.
6. *Refined surface from excitatory curves and surface.* Incidence curve list can be constructed in $O(s + c)^2$ time. Total vote convolution, aggregation, and interpretation takes at most $O((s + c)tk^3 + d^3)$ time. The maximal surface extraction algorithm runs in time linear with the output size, i.e. at most $O(st)$ time.
7. *Final curves and junctions from refined surfaces.* The surface / surface intersection routine can be embedded in (6), where each intersection between two voxel surfaces takes $O(1)$ time.

In all, the most time consuming part is step (6), because voting is performed on surface *dense* data (given by SMap). Very often a worst-case time complexity analysis as above is very pessimistic. However, in our implementation, for instance, we ignore insignificant values far away from the field centroid in all voting steps, which reduces the average running time for vote aggregation and interpretation (since not every voxel needs to be considered). Using a 50x50x50 voxel array to quantize about 1000 input points, our program runs in about 15 minutes on a Sun Sparc Ultra 1.

7 Results

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

Plane and sphere. A total of 342 data points with normals are sampled from a sphere intersecting with a plane. Figure 13 shows some views of the input data, the extracted intersection curve, and integrated result obtained using our cooperative approach.

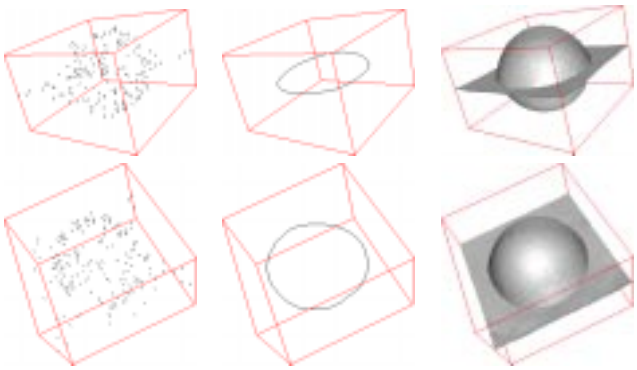

Fig. 13. Plane and sphere. An intersection curve is precisely inferred. The integrated surface description consists of an upper and lower hemispherical surfaces, a square planar surface with a circular hole, and a circular planar surface inside the sphere (not visible).

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 Figure 14.

Triangular wedge. A digitizer is made to sweep over a real triangular wedge and a set of 1266 data points are obtained. 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 Figure 15.

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 Figure 16.

8 Conclusion and future work

In this paper, we describe a general method for inferring coherent surface, curve, and junction from a sparse 3-D data set in which discontinuities are explicitly preserved. The cooperative approach refines initial estimates by using excitatory and inhibitory fields (which are derived from an earlier work [5],

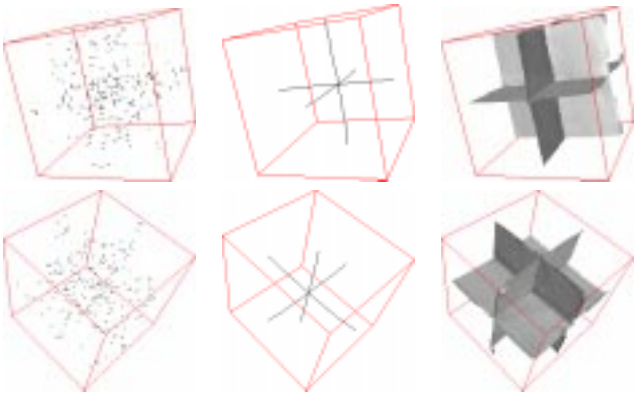

Fig. 14. Three orthogonal planes. There are six maximal curves and one 6-junction which are coherently integrated with twelve maximal planar surfaces. The junction curves are properly localized.

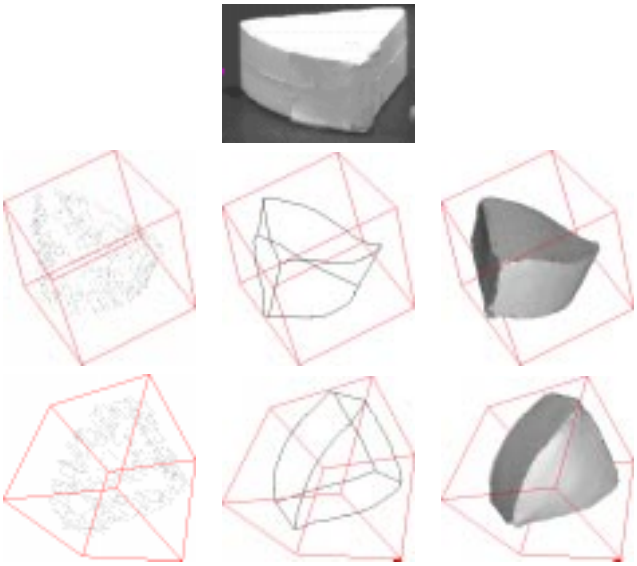

Fig. 15. The surface of a triangular wedge is sampled using a digitizer. The set of points obtained is used as input to our program to generate the integrated descriptions of junctions, curves, and surfaces. Six 3-junctions and nine maximal curves, and six maximal surfaces are integrated. The real object is also shown here.

[6]) and novel surface and curve tracing processes that produce maximal surfaces and curves given dense saliency maps. 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. The use of more sophisticated voting schemes, such as tensor voting, to perform more accurate voting is another issue. Together with the scale of analysis, which is pertinent to the size of mask, and the extension of the methodology to other problems, are the topics of our current research effort.

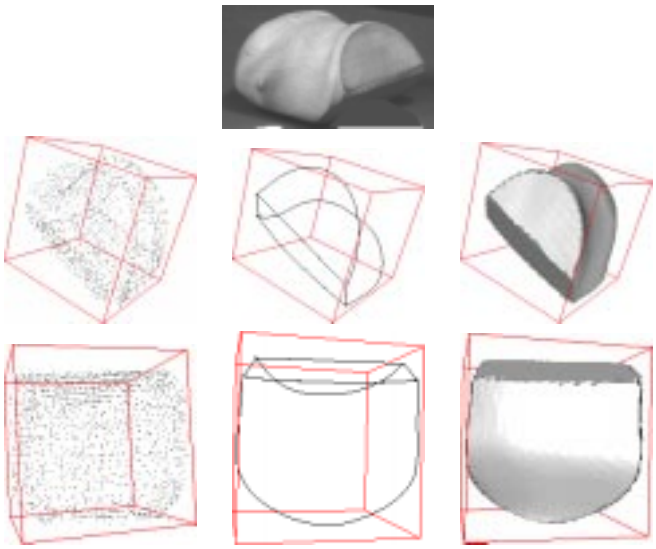

Fig. 16. The surface of a wooden block is sampled using a digitizer. The set of points obtained is used as input to our program to generate the integrated descriptions of junctions, curves, and surfaces. Four 3-junctions and six maximal curves are inferred and integrated with four maximal surfaces, some of them have non-constant curvature. The real object is also shown here.

References

[1] J. D. Boissonnat, "Representation of Objects by Triangulating Points in 3-D space", in *Proc. of the Sixth Intl. Conf. Patt. Recogn.*, pp. 830–832, 1982.

[2] T. E. Boult and J. R. Kender, "Visual Surface Reconstruction Using Sparse Depth Data", in *Proc. Comput. Vision Patt. Recogn.* (Miami Beach, FL), pp. 68–76, 1986.

[3] A. Blake and A. Zisserman, "Invariant Surface Reconstruction Using Weak Continuity Constraints", in *Proc. Comput. Vision Patt. Recogn.* (Miami Beach, FL), pp. 62–67, 1986.

[4] P. Fua and P. Sander, "Segmenting Unstructured 3D Points into Surfaces", in *Proc. Euro. Conf. Comput. Vision* (Santa Margherita Ligure, Italy), pp. 676–680, 1992.

[5] G. Guy and G. Medioni, "Inferring Global Perceptual Contours from Local Features", *Int. J. Comput. Vision*, vol. 20, no. 1/2, pp.113–133, 1996.

[6] G. Guy and G. Medioni, "Inference of Multiple Curves and Surfaces from Sparse Data", *IRIS-USC Technical Report (PhD Thesis)*, University of Southern California.

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

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

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

[10] M. S. Lee, "Inference of Layered Descriptions from Images", *IRIS-USC Technical Report (Research Proposal)*, Institute for Robotics and Intelligent Systems, University of Southern California.

[11] C. Liao and G. Medioni, "Surface Approximation of a Cloud of 3-D Points", *CAD94 workshop* (Pittsburgh, PA), 1994.

[12] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface R Reconstruction Algorithm", *Computer Graphics*, 21(4), 1987.

[13] T. Poggio and F. Girosi, "A theory of networks for learning", *Science*, pp. 978–982, 1990.

[14] R. Szeliski, D. Tonnesen, and D. Terzopoulos, "Modeling Surfaces of Arbitrary Topology with Dynamic Particles", in *Proc. IEEE Comput. Vision Patt. Recogn.* (New York City, NY), pp. 82–85, June 1993.

[15] C. Tang and G. Medioni, "Extremal Surface and Curve Algorithms", *IRIS-USC Technical Report*, University of Southern California.

[16] D. Terzopoulos and D. Metaxas, "Dynamic 3D models with local and global deformations: deformable superquadratics", *IEEE Trans. on Patt. Anal. and Machine Intell.*, vol. 13, no. 7, pp. 91–123, 1991.

[17] D. Terzopoulos and M. Vasilescu, "Sampling and reconstruction with adaptive meshes", in *Proc. IEEE Comput. Vision Patt. Recogn.* (Lahaina, Maui, HI), June 1991, pp. 70–75.

[18] 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.

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

[20] M. Vasilescu and D. Terzopoulos, "Adaptive Meshes and Shells: Irregular Triangulation, Discontinuities, and Heretical subdivision", in *Proc. Comput. Vision Patt. Recogn.*, pp. 829–832, 1992.