Inference of Surfaces, 3D Curves, and Junctions From Sparse, Noisy, 3D Data

Gideon Guy and Gérard Medioni, *Senior Member, IEEE*

Abstract—We address the problem of obtaining dense surface information from a sparse set of 3D data in the presence of spurious noise samples. The input can be in the form of points, or points with an associated tangent or normal, allowing both position and direction to be corrupted by noise. Most approaches treat the problem as an interpolation problem, which is solved by fitting a surface such as a membrane or thin plate to minimize some function. We argue that these physical constraints are not sufficient, and propose to impose additional perceptual constraints such as good continuity and "cosurfacity." These constraints allow us to not only infer surfaces, but also to detect surface orientation discontinuities, as well as junctions, all at the same time. The approach imposes no restriction on genus, number of discontinuities, number of objects, and is noniterative. The result is in the form of three dense saliency maps for surfaces, intersections between surfaces (i.e., 3D curves), and 3D junctions, respectively. These saliency maps are then used to guide a "marching" process to generate a description (e.g., a triangulated mesh) making information about surfaces, space curves, and 3D junctions explicit. The traditional marching process needs to be refined as the polarity of the surface orientation is not necessarily locally consistent. These three maps are currently not integrated, and this is the topic of our ongoing research. We present results on a variety of computer-generated and real data, having varying curvature, of different genus, and multiple objects.

Index Terms—Segmentation and feature extraction, human visual perception issues, isosurface extraction.

1 Introduction

The human visual system can perform an amazingly good job of perceiving surfaces from a set of 3D points. It cannot only infer surfaces, but also segment the scene into objects, and detect surface orientation and discontinuities. This can be demonstrated by displaying a sequence of views corresponding to a set of 3D points from different viewpoints. Consider, for example the scenario in Fig. 1. First, a sparse set of points is sampled from the plane and the sphere. Next, these 3D points are presented to the viewer as a sequence of projections. Human have no problem perceiving the geometric shapes and segmenting them.

This is the performance of shape inference we are trying to emulate here. We argue that traditional interpolation techniques are unable to account for such performance, except in much restricted scenarios (e.g., one single smooth surface with a single visible sheet).

Our approach consists of using perceptual constraints between various 3D primitives in space in order to group these features, and to reconstruct the underlying surfaces. This is achieved by a nonlinear voting process implemented as a convolution of the input features with a predefined mask (later referred to as the Diabolo Field) resulting in a dense saliency map of the space. Such a map holds high values for locations in space which are strong candidates for surfaces. Also, at each such location, a normal to the predicted surface is available. Space curves and 3D junctions can be readily derived from the same map, as we shall show later. The process is noniterative, parameter free, and allows for any number of objects in the scene, each with any genus. An earlier 2D work by the authors [9], [11], uses a similar, but simplified method to extract curves in intensity images.

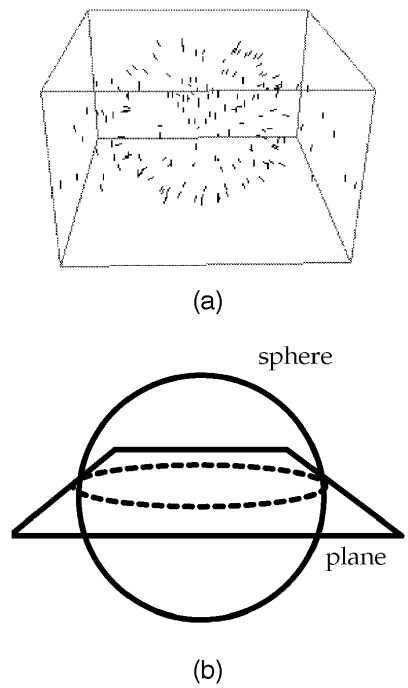


Fig. 1. (a) A sparse set of 3D points (with normals) are sampled from a plane intersecting a sphere. (b) The dotted circle represents the intersection contour and is not explicit in the data.

- *G. Guy is currently with Intelligent Computer Solutions Inc.*
- *G. Medioni is with the Institute for Robotics and Intelligent Systems, 204 Powell Hall of Engineering MC0273, University of Southern California, Los Angeles, CA 90089-0273. E-mail: medioni@iris.usc.edu.*

Manuscript received 25 Mar. 1996; revised 25 Aug. 1997. Recommended for acceptance by B.C. Vemuri.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 105689.

We start by briefly reviewing some of the existing work on this problem, then describe our approach, and finally show results on complex data.

2 Previous Work

Much work has been done in fitting surfaces to clouds of points (for a detailed discussion, see [14]). The three main trends are physics based, functional minimization, and computational geometry based approaches.

The deformable model approach (first proposed by Kass et al. [13] for 2D, and in [23] for 3D) attempt to deform an initial shape so that it fits a set of points through energy minimization. Boult and Kender [3] addressed the sparse data problem specifically. They analyzed and compared four theoretical approaches, and demonstrated a method using minimization over Hilbert spaces. Poggio and Girosi [18] formulated the single-surface approximation problem as a network learning problem. Blake and Zisserman [4] addressed similar problems dealing explicitly with discontinuities. Others (e.g., [20], [21], [22], [24]) also studied similar problems.

Fua and Sander [6] have proposed a local algorithm to describe surfaces from a set of points. Szeliski et al. [19] have proposed the use of a physically-based dynamic local process evolving from each data point, and subject to various forces. They are able to handle objects of unrestricted topology, but assume a single connected object.

The physics based approaches ([21], [22], and [23]) rely on the Governing Equation which is typically used to describe multiparticle systems. The model itself is thought of either as a mesh-like surface composed of a set of nodes with springs connecting them, or as a surface represented by an implicit function. The model (or initial surface) is in equilibrium without external forces exerted on it. The input data points are now attached to their counterparts on the model by springs. This interrupts the equilibrium, and requires the system to converge to a new steady state, which is considered to be the desired surface. In addition to the external force from the data points, there are two more forces in this dynamic system. One is from the stiffness of the spring (or the surface), and the other is the damping for dissipating the energy of the whole system. Some approaches based on this might need the correspondence relation between the collected measurements and the points on the model, which are not always available.

The functional minimization approach (for example, [12] and [14]) always starts with an initial surface, which is then made to fit the data. An energy function is associated with the surface which indicates the present energy of the surface. The energy comes mainly from

1) the smoothness constraint imposed on the fitting surface and
2) the distance between the fitting surface and the data.

A numerical method for function minimization is then applied to conform the fitting surface to the data by reducing the energy of the fitting surface. The underlying surface of the collected data is obtained when the function reaches the minimum.

The algorithms in the computational geometry category (for example, [2] and [7]) treat the collected data as vertices of a graph. The graph is constructed by adding edges between nodes, based on estimated local properties. The algorithms in this category usually produce results in a polyhedral form where each face is a triangle, and the vertices on the polyhedron are the input data points themselves. Using local information only, computational geometry based approaches cannot handle noise. They can, however, describe complex objects with any topology.

The above methods are computationally expensive as an iterative process takes place. They also suffer from one or more of the following problems:

- Only one genus-zero object—This is true to all deformable models approaches.
- A single 2 1/2-D surface can be described at any one tim—Spatial ordering is known. Such methods fail when true 3D data is present.
- Smoothing—Surface boundaries and discontinuities are usually smoothed out as most methods do not handle specifically surface boundaries.
- Outlier noise—Noisy data can severely interfere with the convergence to the correct surface. Methods requiring that the resulting surface to pass through all input data points collapse when noise is present.

3 Overview of Our Approach

We consider inputs in three forms:

1) 3D points,
2) short curve segments with partial orientation data, and
3) 3D points with surface normal information (patches).

In all cases, the input consists of a sparse set of data. In our current implementation, these primitives are quantized in a 3D array.

We consider (3) as the “basic” case. When it is not the case, we preprocess the data to reduce them into our “basic” case by estimating the normals for each input point or sites. As we shall see, the process of normal estimation does not differ from the main process of recovering surfaces from the data, which involves a convolution with a vector field that capture perceptual constraints. The perceptual constraints we wish to locally enforce for the desired surfaces are: Cosurfacity (the 2D counterpart is cocurvilinearity), proximity, and constancy of curvature.

We encode these constraints into a vector field, called the Diabolo Field, whose design is detailed in Section 4. Such a field, when aligned with an input site, associates a preferred direction and strength to every voxel in a large volume of space around the input site.

By aligning the field with each input site, we produce, at each voxel location, a collection of vector votes. This complex information is then compressed into the second order moments of the vector collection, graphically represented by an ellipsoid, or equivalently, by three orthogonal 3D vectors. These three vectors are then combined and interpreted as three independent saliency volume maps corresponding to surfaces, curves and junctions, as explained in Section 5.3.

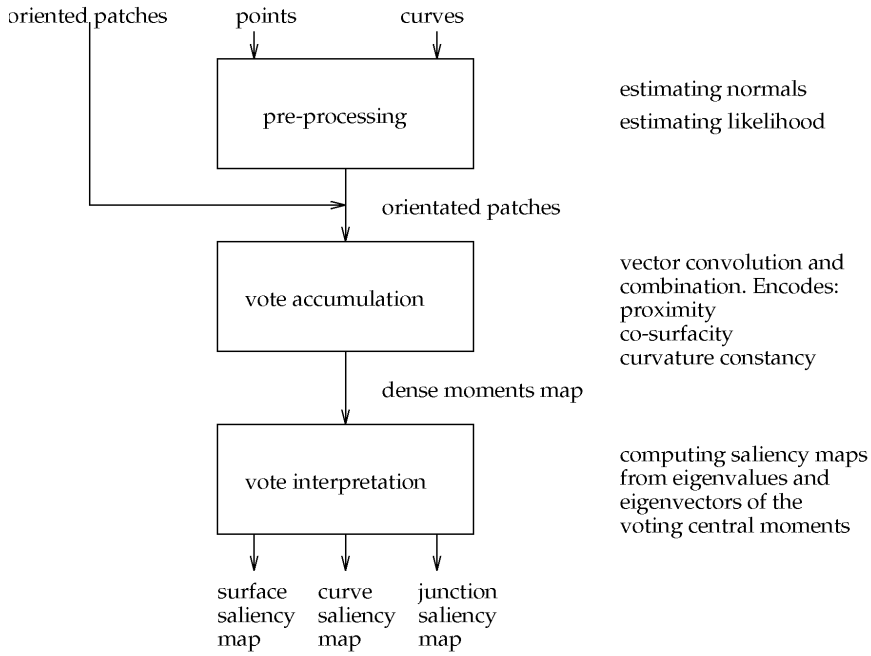


Fig. 2. Flowchart of our system.

A subsequent step of integration of the information contained in these three maps is under investigation. For each of these maps, we extract the maximal sets (surfaces, curves and points) with subvoxel precision using an appropriately modified "marching" algorithm, as explained in Section 5.4.

The flowchart of the approach is described on Fig. 2. It, however, does not include the off-line computation of the fields mentioned above. They are performed only once at the beginning. We shall discuss them in the next section.

4 The Design of the Fields

With no loss of generality, we assume that the input consists of separate primitives, or small patches. All samples are placed on a regular grid, which allows for a much simpler implementation. We next describe the rationale behind the shape of the fields.

4.1 The Diabolo Field

Given a patch at the origin with a known normal (N), we ask the following question: for a given point P in space, what is the most likely normal (at P) to a surface passing through P, and tangent to the original patch? (Fig. 3 illustrates this situation.) We claim that a circular continuation between the patch and point P is the most appealing one, since it keeps the curvature constant along the hypothesized circular arc. The most likely normal then is the normal to that arc at P. Also, symmetry considerations dictate that it lies on the same plane as the original normal N.

We set the length of these normal vectors to be inversely proportional to the distance of the point from the patch, and to the curvature of the underlying circular arc. This effectively encodes both proximity and the lower curvature constraints, and represents the likelihood of a surface passing through that point. It is possible to tune the likelihood values, so as to optimize the behavior of the system. In our current implementation, we use a Gaussian decay function that seems to work well on all examples. This predefined mask is named a Diabolo Field (DF) because of its resemblance to the "Diabolo," a device used by jugglers, as shown in Fig 4. In spherical coordinates, the Diabolo Field thus takes the following form:

$$\overline{DF}(r,\varphi,\theta) = e^{-Ar^2} e^{-B\varphi^2} \quad (1)$$

where A encodes the decay due to proximity, and B the decay due to higher curvature. The parameters A and B were selected empirically (A = 0.003, B = 2.85) based on some simple scenarios so that the desired response is achieved, and are kept unchanged throughout all of our experiments (see [10] for details).

The Diabolo Field is illustrated in Fig. 5, and has the general shape of nested half spheres. Notice that for points which are farther than 90° along the spherical surface, a circular arc is no longer the most likely continuation, and in fact we assign a zero value to field elements there. This means that a given patch in space does not vote inside a 45° cone whose axis aligns with the normal to that patch.

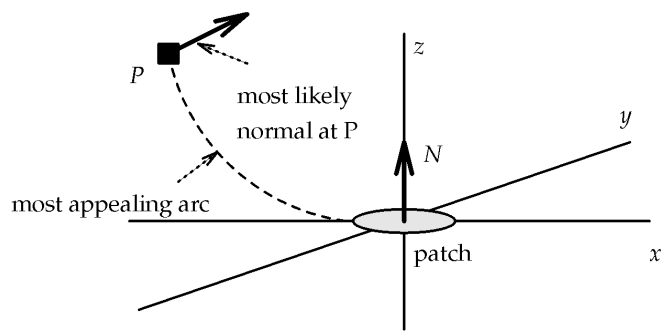


Fig. 3. What is the most likely normal to a surface passing through point P and at the same time tangent to the patch at the origin?

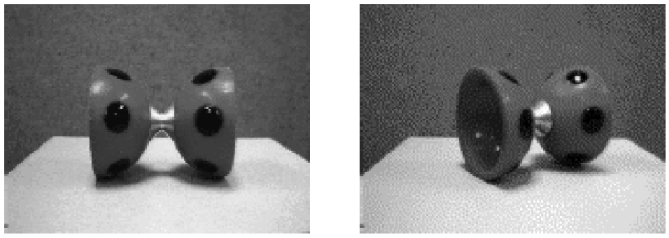


Fig. 4. Two views of a juggler's diabolo.

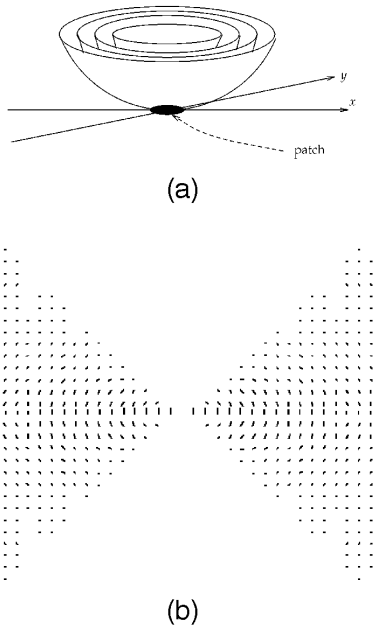


Fig. 5. (a) The general shape of the Diabolo Field. The lower part is a mirror image of the top and omitted here for clarity. Field vectors (not shown) are normal to the "bowl" surfaces shown. (b) A cross-section through the $y = 0$ plane, with actual voting vectors.

4.2 The 3D Curve Segment Field

The curve segment field is designed to aid in the pre-processing step when line segments that are known to be tangent to the surfaces are present in the input data. Such data is typical when using a 3D digitizer in its sweep mode, or when surface markings are matched in a stereo pair. Matched (true or limb) edges in a stereo pair can also supply input in the appropriate format. Clearly, having some information about the site is better than only having the location as is the case in the completely nonoriented case. We thus refer to this type of input as the partially-oriented input.

Again we ask the question: What is the most likely surface to pass through a point P in space and have the segment at the origin lying on it? The answer is very simple. A segment and a point in space define exactly one plane. And since a plane is the most likely surface in terms of the perceptual constraints, it is also the most likely to appear.

A practical way of constructing this field is to take the Diabolo Field and convolve it with a multidirectional patch. This last operation is similar to revolving the Diabolo Field around itself along the x (or y) axis (referring to notation in Fig. 6). By symmetry considerations, it is simple to show that the resulting field will have the correct orientations

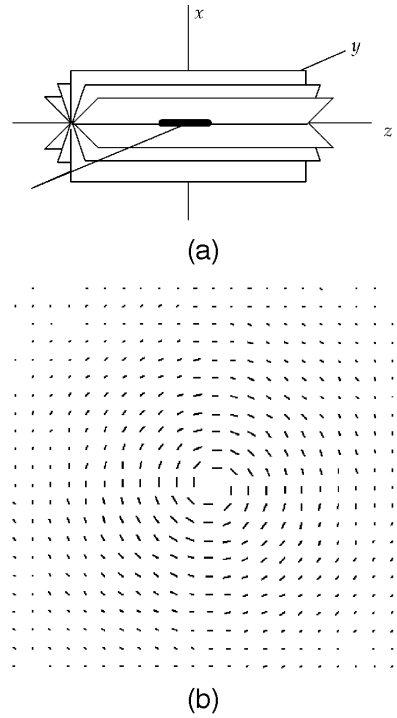


Fig. 6. The general shape of the curve segment field. (a) All planes go to infinity, with diminishing strength. (b) A cross-section at $z = 0$. The field elements are normals to the drawn planes.

everywhere in space (as shown in Fig. 6). This construction also determines the strength values at every site of the field.

4.3 The 3D Point Field

The 3D Point field is necessary for the pre-processing step to handle nonoriented input samples. Here, we ask the question: Given a point in space and the origin, what is the most likely surface to pass through these two points? The answer is that a family of planes going through the two points are all equally likely (in the absence of any other cue, there is no reason to hypothesize a curved surface). That is, at each point in space, many normals are equally likely (see Fig. 6). They all lie on a plane perpendicular to the line formed by the point in question and the origin. We thus choose to describe the contribution of all these normals with a single 3D vector pointing in the direction of the above line (thick arrow in Fig. 6). This treatment results in a simple radial field with diminishing strength.

We later show how such a voting vector is applied to estimate normals for partially- and nonoriented input data.

5 Implementation of the Algorithm

We now know how to locally impose the constraints of cosurfacity, proximity, smoothness and low curvature, by encoding them into a vector field. We show how to aggregate these local inferences.

The algorithm consists of four stages (for oriented inputs):

- Voting
- Representation
- Interpretation
- High-level feature extraction

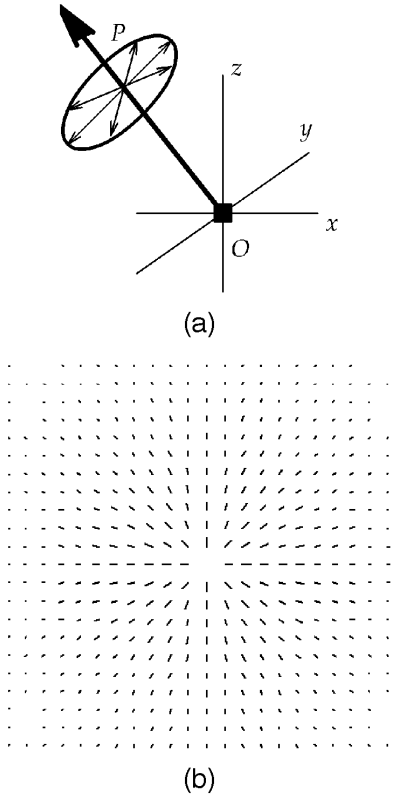


Fig. 7. (a) All normals (thin arrows) at point P are equally likely. We choose to represent all of them with one vector (thick arrow) perpendicular to the plane they all lie on. (b) A cross-section of the Point Field at $y = 0$.

5.1 Vector Convolution and Vote Aggregation

The computation of the Saliency maps is implemented as a vector convolution with the above fields. The resulting map is then a function of a collection of fields, each oriented along a corresponding short input normal in 3D. Each site accumulates the "votes" for its own preferred orientation and strength from every other site in the volume. These values are combined at every site as described next.

5.2 Representation at Each Voxel

At this stage in the algorithm, each voxel in the array has collected votes from every input site. We need to compute some measure of coherence (or agreement) in this collection of vectors, in terms of their orientation. If many of the vectors point in (roughly) the same direction, we say that there is a high level of agreement in the votes, and for that reason, the likelihood of a surface passing through that voxel is high. We should also consider the total weight of the voting vectors.

In practice, we treat the contributions to a voxel as being vector weights, and compute central moments of the resulting system. Computing the second order moments of the system is equivalent to finding a 3D ellipsoid having the same moments and principal axes. Such a physical model acts as an approximation to a majority vote (or a statistical mode), and behaves in the desired way, giving both the preferred direction and a measure of the agreement.

Outlier locations are likely to produce an incoherent set of votes, which, as we show later, produces a low saliency measure.

5.3 Vote Interpretation

It is now that we actually reason about the data in hand. We claim that the eigenvalues and eigenvectors associated with the moments at each site can be used to estimate saliency and predicted normals.

For each site in our 3D array we decompose the 3×3 variance-covariance matrix, as expressed in (2), into (3), where λ_{max}, λ_{mid}, and λ_{min} represent the three sorted eigenvalues of the system, m_{abc} denote the corresponding sum of moments at each voxel, and the Vs denote the corresponding eigenvectors of the system. Such decomposition will always yield real, nonnegative eigenvalues since the matrix is positive semidefinite:

$$\begin{bmatrix} m_{200} & m_{110} & m_{101} \\ m_{110} & m_{020} & m_{011} \\ m_{101} & m_{011} & m_{002} \end{bmatrix} \tag{2}$$

which is equal to

$$\begin{bmatrix} V_{min} \\ V_{mid} \\ V_{max} \end{bmatrix}^T \begin{bmatrix} \lambda_{min} & 0 & 0 \\ 0 & \lambda_{mid} & 0 \\ 0 & 0 & \lambda_{max} \end{bmatrix} \begin{bmatrix} V_{min} \\ V_{mid} \\ V_{max} \end{bmatrix} \tag{3}$$

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, as explained next.

λ_{max}, λ_{mid}, and λ_{min} are related to the number of votes received. By itself, however, this raw value does not capture the desired saliency, as it does not capture the agreement

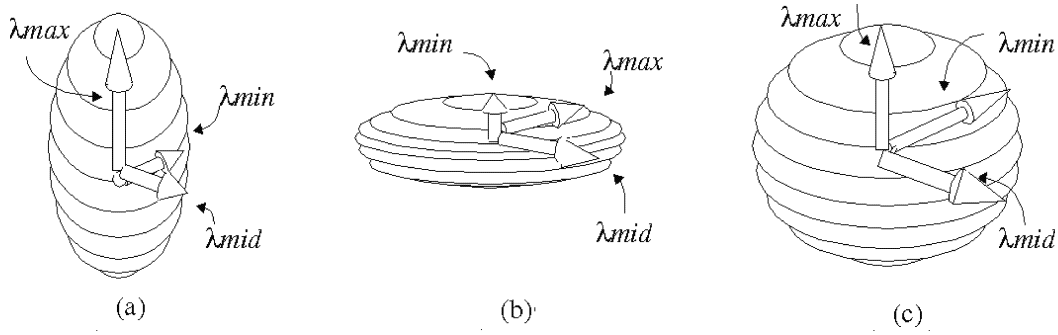


Fig. 8. The three important voting ellipsoids. (a) $\lambda_{max} \gg \lambda_{mid} \approx \lambda_{min}$, high agreement in exactly one direction (a surface). (b) $\lambda max \approx \lambda_{mid} \gg \lambda_{min}$, high agreement in exactly two orientations (an intersection, or 3D curve). (c) $\lambda_{max} \approx \lambda_{mid} \approx \lambda_{min}$, votes are coming from all directions (a 3D junction).

between votes. In order to interpret these three values, it is useful to consider what happens on a smooth surface, at the edge between two surfaces, and at a junction between three or more surfaces, as illustrated in Fig. 8a.

On a smooth surface, the votes produce high agreement around one direction, leading to the situation depicted in Fig. 8a, where $\lambda_{max} \gg \lambda_{mid}, \lambda_{min}$.

Along the curve bounding two surfaces, two of the eigenvalues are high, and one small, as in Fig. 8b, leading to $\lambda_{mid} \gg \lambda_{min}$.

Finally, at a junction of two or more curves, all three values are similar, as in Fig. 8c.

Based on these observations, we propose to define three voxel maps defining the surface, curve, and junction *saliencies*, respectively. Each voxel of these maps has a two-tuple $(s, \bar{v})$, where s is a scalar and $\bar{v}$ is a unit vector:

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

Note that these three maps are dense saliency maps, which need to be processed (maxima extraction) to produce the derived features, as explained in the following (Section 5.4).

5.3.1 Preprocessing for the Nonoriented cases

The basic case we are considering is that a normal is available at each input site. 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 (point or curve segment), and interpreting the resulting votes, but only at the original input sites, making this step computationally inexpensive.

As a result of this step, each input site now holds a normal, obtained as the eigenvector V_{max} corresponding to λ_{max}.

5.4 High-Level Feature Extraction

We now have, at each voxel site, the three saliency measures derived from the three eigenvalues and their associated eigenvectors. We describe below the procedure to extract the salient features, corresponding to local maxima of the three saliency maps.

5.4.1 Junction Saliency

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

Our system is limited in resolution, so we do not expect to detect two close by junctions, and therefore inhibit a neighborhood of five voxels around each strong local maximum. Fig. 24d depicts the 3D junctions inferred from the input data (Fig. 24b).

5.4.2 Curve Saliency

Each voxel in the curve map holds a two-tuple $(s, \bar{v})$, where the (scalar) saliency measure s is

$$s = \lambda_{mid} - \lambda_{min} \tag{4}$$

and $\bar{v} = V_{min}$. Since $\bar{v}$ is aligned with the tangent, we will denote it by $\bar{t}$ in this section.

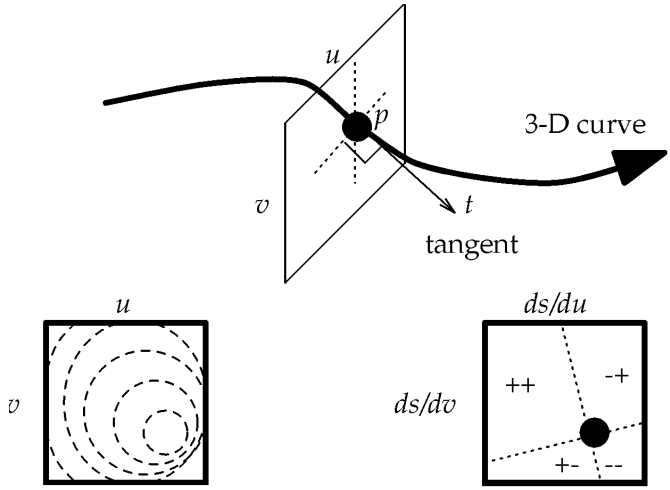


Fig. 9. A maximal curve. (a) $\bar{g}$ is projected onto the plane perpendicular to the tangent to the curve. This projection is achieved by the first cross product of (7). (b) After repositioning the derivatives (done by the second cross product), a change in signs of derivatives in both u and v directions indicates a curve is passing through that face.

Note that the knowledge of $\bar{t}$ at every voxel is essential (otherwise the problem is meaningless). Simply computing local extrema at each voxel location is a very coarse approximation which produces a set of points, rather than a set of connected curves.

We first study the continuous version of the problem of extracting salient curve from a curve map, in which $(s, \bar{t})$ is defined for every point p in 3D space. A point p with $(s, \bar{t})$ is on a *maximal curve* if any displacement from p on the plane normal to $\bar{t}$ will result in a lower s value. In other words, a point p lies on a maximal curve if at point p the following differential property holds:

$$\frac{ds}{du} = \frac{ds}{dv} = 0 \tag{5}$$

where u and v define the plane normal to $\bar{t}$. 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 $\bar{g}$ computed on the strengths of adjacent saliencies,

$$\bar{g} = \begin{bmatrix} \frac{ds}{dx} \\ \frac{ds}{dy} \\ \frac{ds}{dz} \end{bmatrix} \tag{6}$$

and define $\bar{q}$ as

$$\bar{q} = (\bar{t} \times \bar{g}) \times \bar{t} \tag{7}$$

By construction, $\bar{q}$ is the projection of $\bar{g}$ onto the plane normal to $\bar{t}$ (Fig. 9). Therefore, a maximal curve is the locus of points for which $\bar{q} = \bar{0}$.

In (discrete) implementation, we need to deal with the fact that the vector function is digitized and quantized, rather than continuous. We therefore need to compute an approximation for $\bar{g}$ and $\bar{q}$. Then, it is possible to generate a subvoxel linear approximation of the true location for which $\bar{q} = \bar{0}$ (if it exists) for each face of a voxel, using an adapted "Marching Squares" algorithm (a 2D version of Marching Cubes algorithm [8]).

Each voxel in the curve map with coordinates (i, j, k) holds ($s_{i,j,k}$, $\overline{t_{i,j,k}}$). The discrete version of $\overline{g}$, denoted by $\overline{g_{i,j,k}}$, is defined by the first order saliency difference, i.e.,

$$\overline{g_{i,j,k}} = \begin{bmatrix} s_{i+1,j,k} - s_{i,j,k} \\ s_{i,j+1,k} - s_{i,j,k} \\ s_{i,j,k+1} - s_{i,j,k} \end{bmatrix} \tag{8}$$

We can simply define the corresponding discrete $\overline{q_{i,j,k}}$ for each voxel as

$$\overline{q_{i,j,k}} = \left(\overline{t_{i,j,k}} \times \overline{g_{i,j,k}}\right) \times \overline{t_{i,j,k}} \tag{9}$$

Therefore, the set of all $\left\{\overline{q_{i,j,k}}\right\}$ defined for all voxels constitutes a *vector* array which can be processed by the adapted "Marching Squares" algorithm, as described below.

We examine the six faces of each voxel. We assume that a curve will intersect exactly two of the faces if it goes through the voxel. To each of the four vertices of each face we attach two signs of the corresponding element of (9). There are 256 possibilities, but only sixteen of them indicate a curve is going through that face. So for example, for a face that is parallel to the z = 0 plane we assign $\left(\overline{q_{i,j,k}}\right)_x$ and $\left(\overline{q_{i,j,k}}\right)_y$, the x and y component of $\overline{q_{i,j,k}}$, to each of the vertices, and examine their sign. Fig. 10 shows some of the more common cases. It can be seen that for a curve to pass through a given face, all four sides need to have a zero-crossing somewhere along them. This requires the signs to alternate as we follow the boundary of the face.

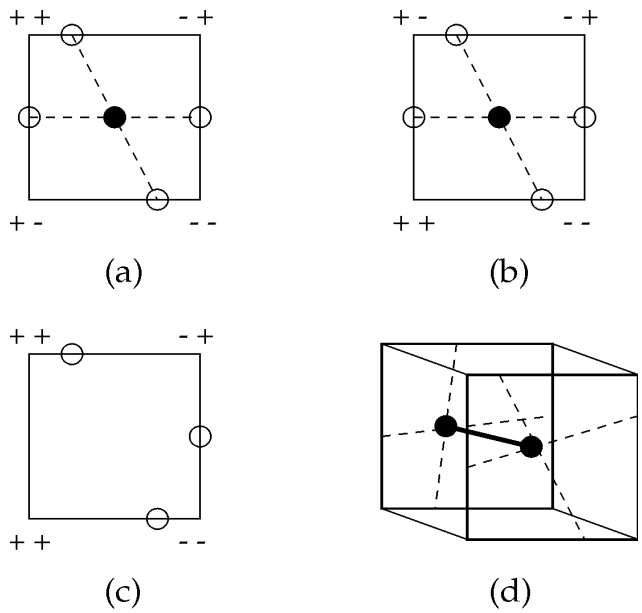


Fig. 10. Discrete version of the maximal curve extraction. (a) and (b) All four sides have a zero-crossing (denoted by the hollow circle) and the curve passes through the intersection of the lines connecting opposite zero-crossings. (c) Only three zero-crossings are found, no curve passes through that face. (d) A curve passes through two faces of a voxel.

An important issue in ensuring the consistency of the evaluation of the four vertices is the *alignment* of the normal vectors. Recall that the front-end of the system (the one that produces the saliency map) cannot ensure that all tangents (or normals) belonging to an underlying curve are oriented consistently along the curve. Thus, the marching algorithm must locally align all the neighboring tangent vectors before applying (9) to them. The alignment involves testing the sign of the dot product of all vectors against an arbitrary vector, and flipping the vector if the sign is negative.

5.4.3 Surface Saliency

Each voxel of the surface map holds a two-tuple (s, $\overline{v}$), where

$$s = \lambda_{max} - \lambda_{mid} \tag{10}$$

and $\overline{v}$ is the unit vector V_{max}. Since $\overline{v}$ aligns with the normal, we will denote it by $\overline{n}$ in this section.

Note again that the knowledge of $\overline{n}$ is essential, and that extracting local maxima at each voxel produces a thick set of points, not a surface.

As before, we first study the continuous version of the problem of salient surface extraction, in which (s, $\overline{n}$) is defined for every point p in 3D 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 following differential property holds:

$$\frac{ds}{d\overline{n}} = 0. \tag{11}$$

This definition involves the detection of zero-crossing on the line aligned with $\overline{n}$. We therefore compute this projection by defining a scalar q which is the dot product of $\overline{n}$ and $\overline{g}$:

$$q = \overline{n} \cdot \overline{g}, \tag{12}$$

where $\overline{g}$ was defined earlier by (6). (Here we have a simpler formulation than in the curve case since q is a scalar.) Therefore, a maximal surface is the locus of points for which q = 0.

As before, in implementation, we need to compute an approximation for $\overline{g}$ and q. Each voxel in the surface map with coordinates (i, j, k) holds ($s_{i,j,k}$, $\overline{n_{i,j,k}}$). $\overline{g_{i,j,k}}$ was defined in (8). So, $q_{i,j,k}$ for each voxel is simply

$$q_{i,j,k} = \overline{n_{i,j,k}} \cdot \overline{g_{i,j,k}}. \tag{13}$$

Therefore, the set of all $\{q_{i,j,k}\}$ constitute a *scalar* array, which can be processed directly by the Marching Cubes algorithm [8] to extract a zero-crossing surface which corresponds to the locus of points for which q = 0.

We illustrate the situation in Fig. 11 and Fig. 12. Fig. 11 shows a slice cut from the surface map of a saddle surface (Fig. 16) parallel to the y-z plane. Each voxel of the surface map holds a 2-tuple (s, $\overline{n}$), see Fig. 11. The rationale behind (12) is depicted in Fig. 12. The gradient is zero at the maximal point, and the dot product described above will change its sign as it moves across the crest line.

As before, alignment of the normal vector is crucial to the success of the maximal surface extraction.

5.5 Noise Tolerance

Our scheme is not sensitive to noise in the form of erroneous features, or localization errors of the measurements, since a nonlinear voting scheme is employed. Also, a priori distribution of noise is expected to be directionally uniform, such that computed orientations are not corrupted. Further

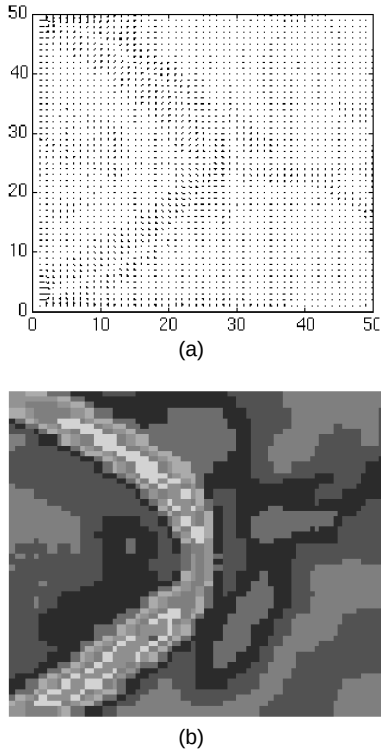


Fig. 11. A slice cut from a surface saliency map along the *y-z* plane. (a) Every voxel has $\overline{n}$, with different level of saliency *s*, shown in (b).

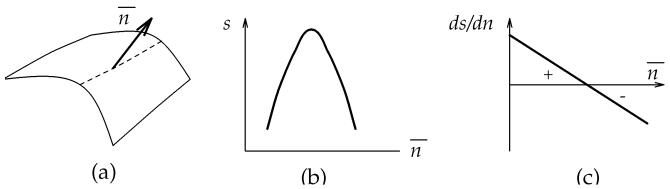


Fig. 12. A maximal surface. (a) A surface patch with its normal, $\overline{n} = V_{max}$, (b) the saliency along the normal, and (c) the derivative of the saliency.

discussion regarding the nonmaximum suppression property can be found in [10]. We present in Section 6 an example in which we add more and more outlier noise to the data, which is sampled from an oval of Cassini (Fig. 18). In this instance (Fig. 19), the algorithm degrades very gracefully, even in the presence of large amount of noise. It is also interesting to note that there are similarities with the Hough transform (see [10] for a more detailed analysis). The major differences are that we do not give here an explicit analytic formulation of the target shape, and that the dimension of the search space is independent of the target shape.

5.6 Complexity

The complexity is $O(n^3k)$ in general, where n is the side size of the volume, and k is the number of available measurements. The complexity of the preprocessing step is $O(k^2)$. Some practical shortcuts can reduce this complexity significantly: a truncated field can be used, since the values of the field get very small away from the center. Also, it is sometimes possible to estimate the minimum true saliency value from similar examples. In such case it is best to threshold the input based on that value, to further crop the input set size.

Fig. 13. A 3D digitizer is used for sampling input data.

Average running time on a Sparc 10 is about five minutes for a $50 \times 50 \times 50$ array with 300 sample points. However, the code is not optimized, nor do we use any of the shortcuts mentioned above.

6 Results

We tested the scheme on synthetic as well as real input data. The real input was acquired by a 3D digitizer (Microsribe Inc.), as shown in Fig. 13.

6.1 Three Planes

The first example consists of three planes positioned in space as shown in Fig. 14. We randomly sample the planes of Fig. 14a and provide position and normal information at each input site. The grid size is $50 \times 50 \times 50$, and each plane has about 60 samples. Fig. 15 shows the final description in terms of space curves, junctions, and polygonal surfaces.

We do not currently produce the expected three planes, as shown in Fig. 14a, but 12 panel surface patches, six curves and one junction, as the three saliency maps are not integrated. This is the focus of our continuing research.

6.2 Saddle Surface

Here we show an example where the input consists of a cloud of nonoriented points, with a considerable amount of noise. We embed 120 samples from a saddle function in noise by adding 50 random points as shown in Fig. 16a. The first phase is to compute normals to the existing input points. This is done by convolving the input with the 3D Point field. The second phase is to perform the standard Diabolo Field convolution. The final result is shown in Fig. 16b. We compute the error surfaces for two cases: noise-free input and noisy input for this saddle function and show the results in Fig. 17. In both cases, the approximating surface is quite faithful.

6.3 Cassini Oval

A total of 263 data points were sampled from another synthetic surface, a peanut mathematically defined as Cassini oval, which is the loci of all points such that the product of the distances from two given (center) points is a constant. The formula is given by:

$$(x^2 + y^2)^2 - 2c^2(x^2 + y^2)^2 - (a^4 - c^4) = 0 \qquad (14)$$

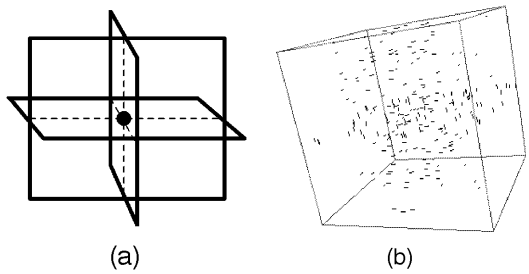


Fig. 14. Three orthogonal planes. (a) A schematic model of input (the dotted lines denote the intersections between surfaces, and the dot is the 3D junction). (b) Projection of input samples.

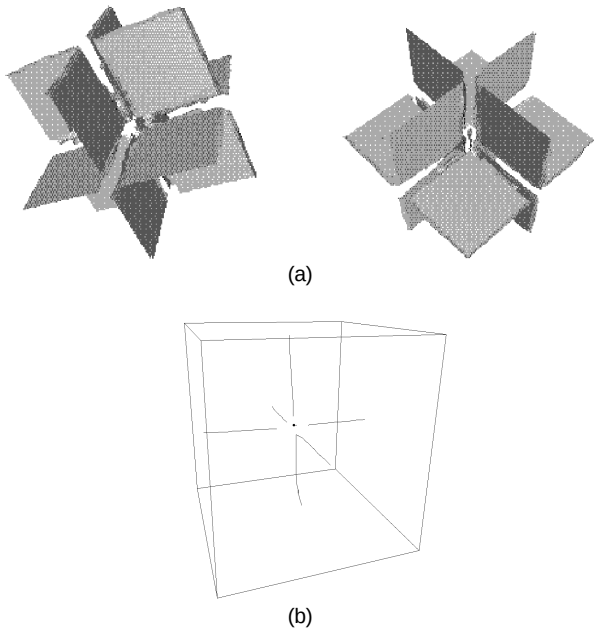


Fig. 15. Final description of the three planes data. (a) Two different views of the 12 reconstructed surfaces. (b) Detected curves and junctions.

where a, $c > 0$ are constants. The final results are shown in Fig. 18.

To illustrate the behavior of the system in the presence of outlier noise, we than add, to $n = 263$ points on the object, random points in increments of n. Note that it is only the injection of $5n$ noise points that the object becomes less salient, and that some parts of the object are still extracted after $6n$ noise points (Fig. 19).

6.4 Torus

Our last synthetic example demonstrates the performance when holes are present in the sampled data set, making it a genus one object. Fig. 20 shows a sampled torus, and two corresponding views of the reconstructed surface.

6.5 Multiple Objects

We generated two scenes to present the capability of our approach to describe multiple objects. The input data of two hemispherical surfaces, one lying inside the other, are shown in Fig. 21. These two surfaces are only six voxels apart. Finally, two separate ellipsoids lying side by side constitute the input data in Fig. 22.

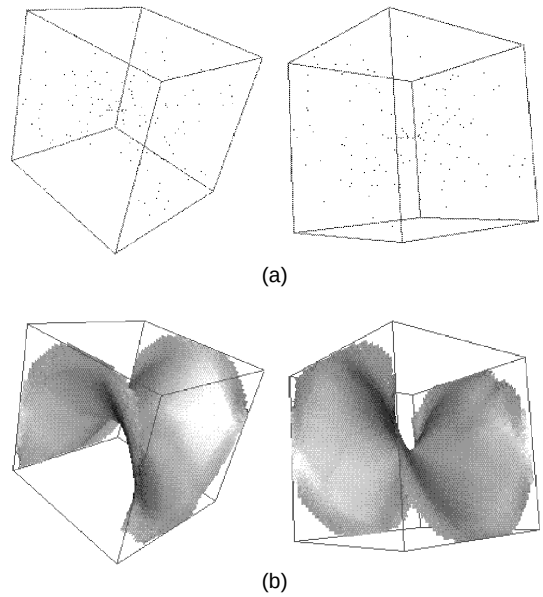


Fig. 16. Recovering a complex nonconstant curvature sheet surface. (a) Two views of the sampled saddle function ($f(x, y) = x^2 - y^2$) about 120 points and 50 random points. (b) Two different views of the reconstructed surfaces.

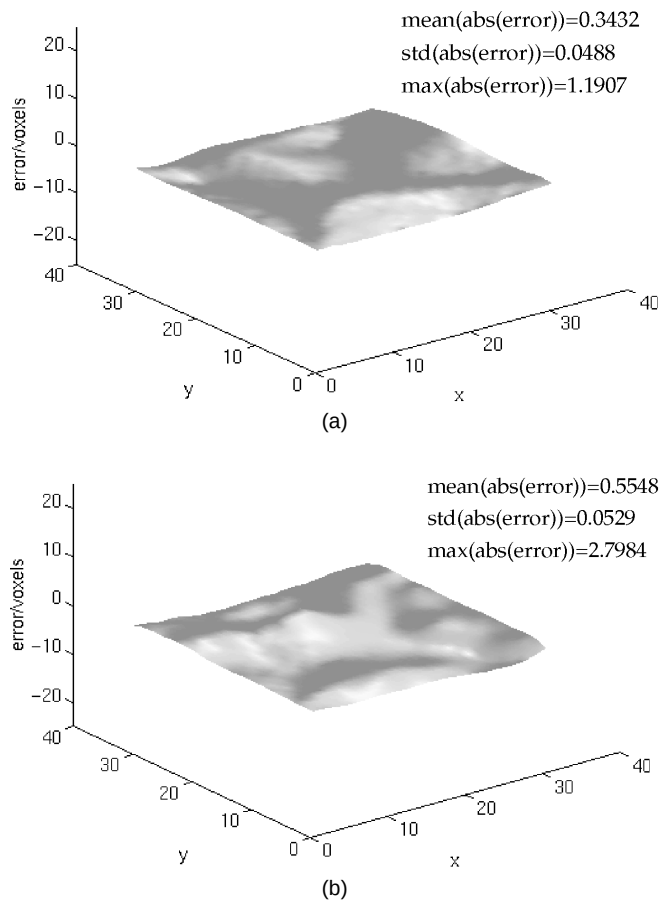


Fig. 17. Error surface for “saddle” (a) with no noise and (b) with noise. The mean, standard deviation, and the maximum absolute error (in voxels) are also shown.

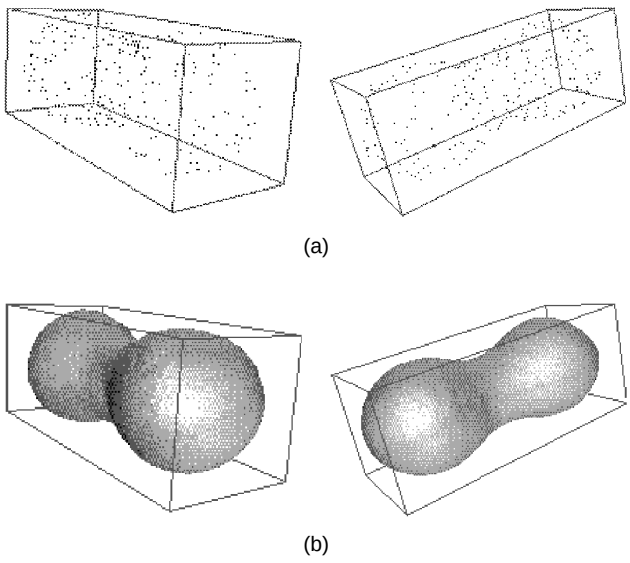


Fig. 18. Peanut-shaped Cassini oval. (a) Two different views of the input (263 points) and (b) resulting surface.

6.6 Real Data

We used a 3D digitizer to digitize a curved wooden block, as shown in Fig. 23. About 170 points are sampled. The tangent vectors were estimated as the vector adjoining pairs of consecutive points. The curve segment field was applied for estimating the normals to the data points, followed by convolution with the Diabolo field.

We also digitized another object, a wedge, and the results are as shown in Fig. 24. Note that we correctly infer the six corners, and the surface orientation discontinuity curves together with the surface. These three independent feature sets should be integrated, which is the focus of our current research effort.

Fig. 25 illustrates the performance of the system when curve segments are presented as input. The preprocessing step here involves convolving the input with the curve segment field, followed by the Diabolo field.

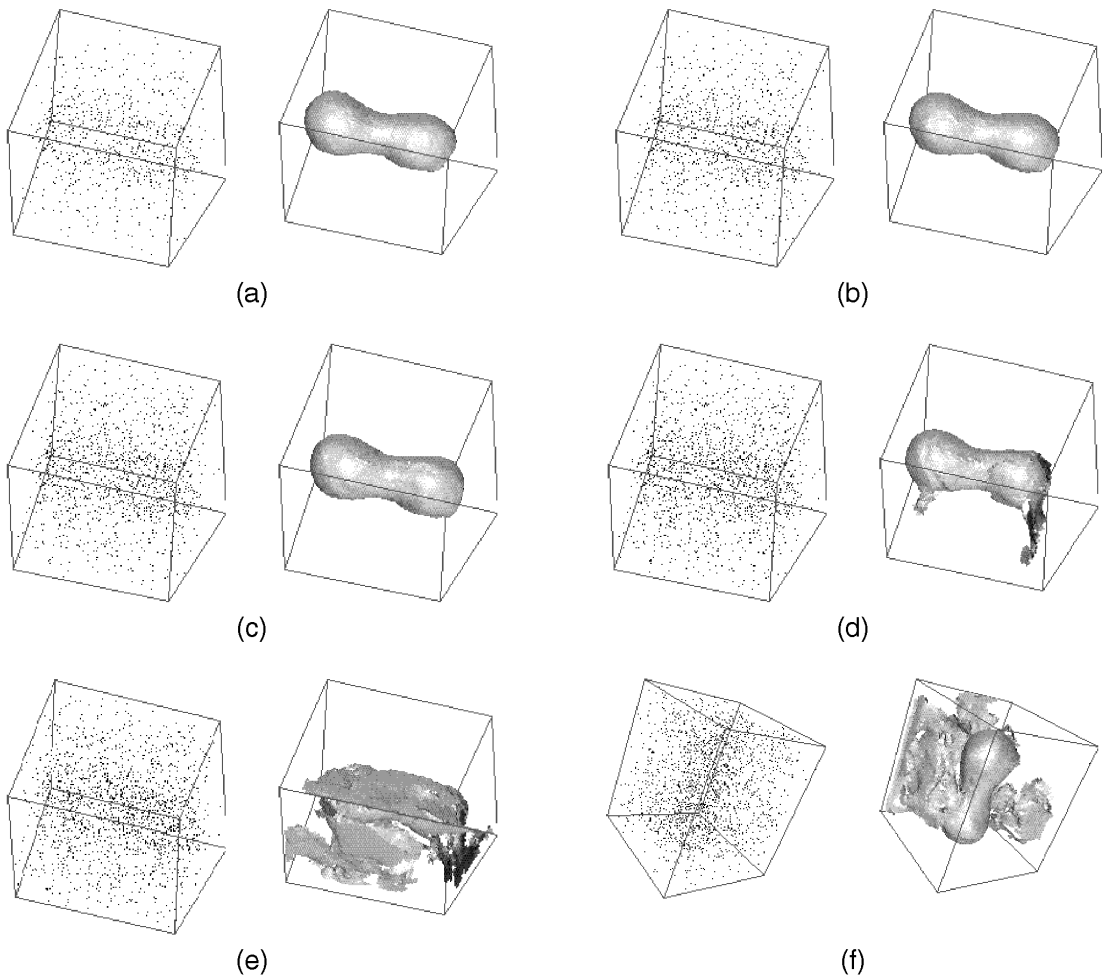


Fig. 19. Graceful degradation of our approach with various amount of additive outlier noises. The example is "peanut," with n = 263 true data points. A view of the input data and a view of the reconstructed surface are shown, (a) with $2n$ additive noises, (b) with $3n$ additive noises, (c) with $4n$ additive noises, (d) with $5n$ additive noises, (e) and (f) two views of the data and result with $6n$ noises.

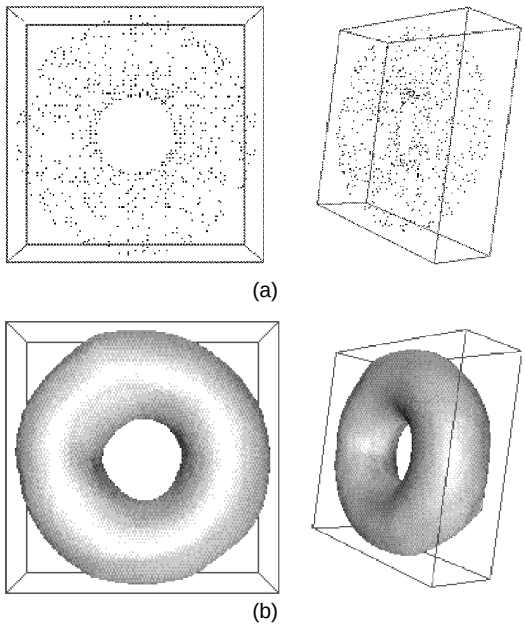


Fig. 20. Torus (genus 1 object). Two different views of the (a) input data (≈ 700 points) and (b) reconstructed surface.

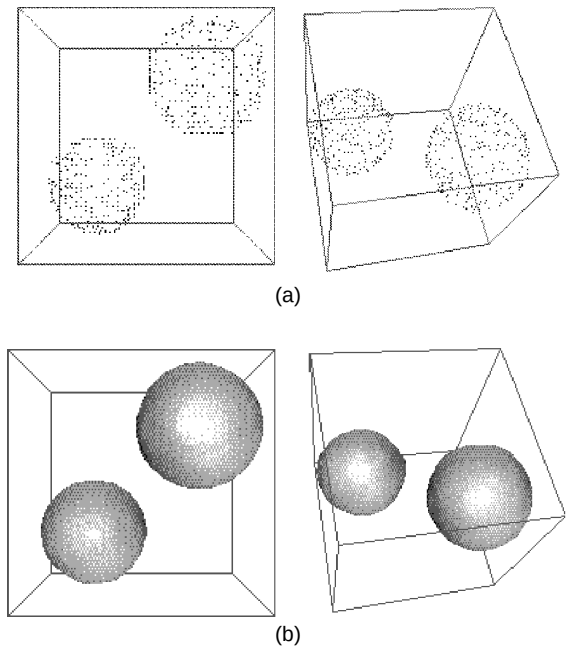


Fig. 22. Dealing with multiple objects. Two views of the (a) input data (≈ 500 points) of two ellipsoids lying side by side and (b) the result.

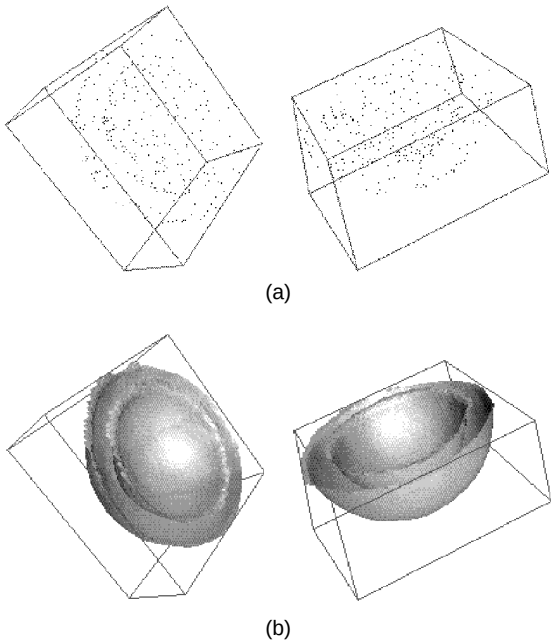


Fig. 21. Dealing with close-by objects. Two views of the (a) input data (≈ 130 points) of two hemispheres with one inside another and (b) the result.

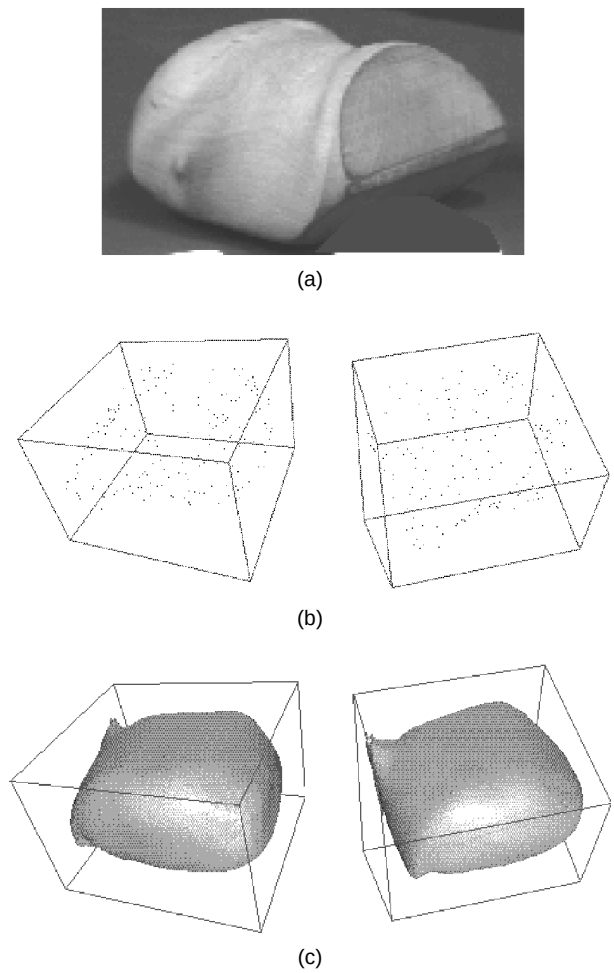


Fig. 23. Curved block. (a) Intensity view, (b) input data (≈ 170 points), and (c) reconstructed surface.

7 Conclusion

We have presented a method to infer surfaces, while detecting intersection between surfaces, and 3D junctions by applying perceptual grouping rules. The method presented is an extension of a 2D approach proposed earlier by the authors, and uses a noniterative algorithm. The method can handle scenes with any number of objects, each having an arbitrary genus number, without any a priori knowledge. In particular, an initial guess is not needed. The current

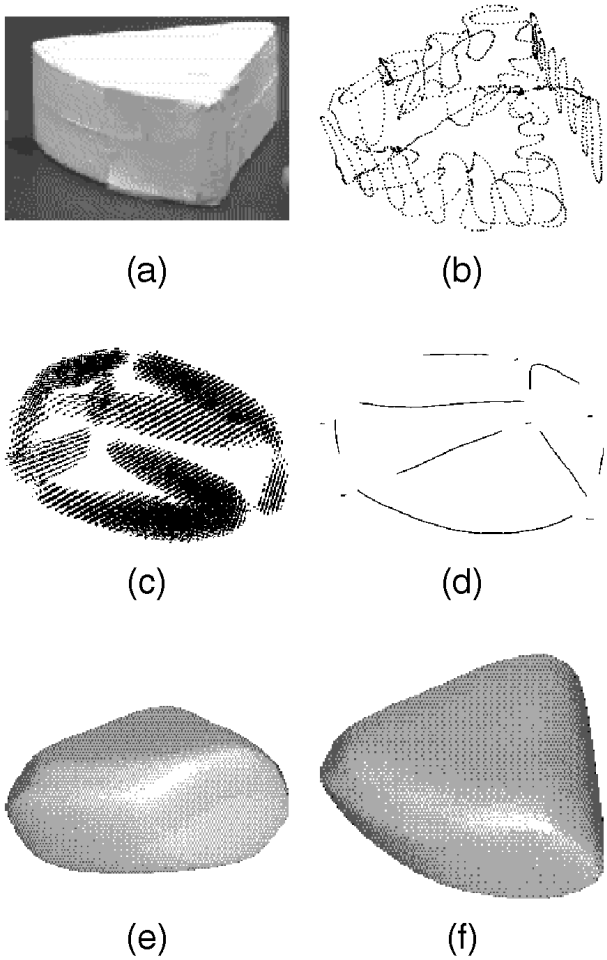


Fig. 24. Wedge. (a) Intensity view, (b) the original sweep, (c) a thresholded version of the curve saliency map, (d) curves and junctions, (e) and (f) two views of the reconstructed surface.

limitations are that the three maps are not integrated. As a result, the curves and the junctions are detected but not well localized, and the surface itself may be disturbed or interrupted by the presence of curves and junctions. Another issue relates to the size of the mask, which is related to the scale of the analysis. These issues, together with extension of the methodology to other problems, is the topic of our current research effort.

ACKNOWLEDGMENT

This work was supported by the U.S. National Science Foundation under Grant IRI-94231441.

REFERENCES

[1] N. Ahuja and M. Tuceryan, "Extraction of Early Perceptual Structure in Dot Patterns: Integrating Region, Boundary, and Component Gestalt," *Computer Vision, Graphics, and Image Processing*, vol. 48, pp. 304-356, 1989.

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

[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.

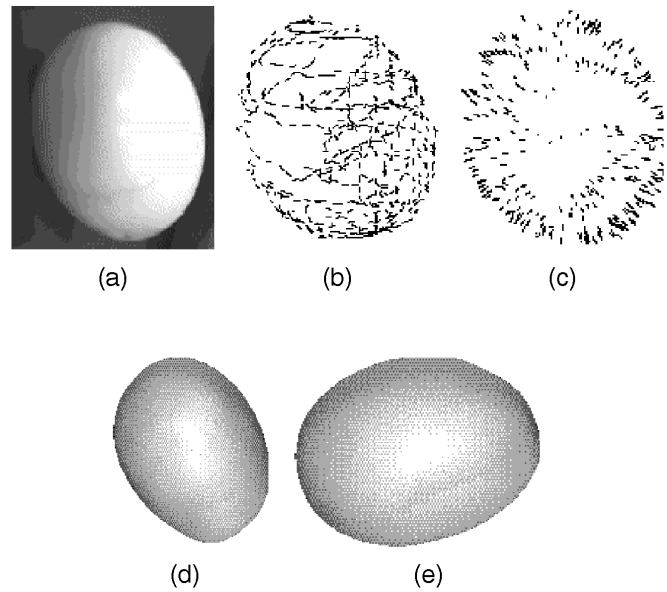


Fig. 25. Wooden egg. (a) Intensity view, (b) input curve segments, (c) estimated normals, (d) and (e) two views of reconstructed surface.

[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] J. Dolan and R. Weiss, "Perceptual Grouping of Curved Lines," *Proc. Image Understanding Workshop*, pp. 1,135-1,145, Palo Alto, Calif., 1989.

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

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

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

[9] G. Guy and G. Medioni, "Perceptual Grouping Using Global Saliency Enhancing Operators," *Proc. Int'l Conf. Pattern Recognition*, pp. 99-104, The Hague, Holland, 1992.

[10] G. Guy and G. Medioni, "Inference of Multiple Curves and Surfaces From Sparse Data," IRIS-USC technical report, PhD Thesis, Univ. of Southern California.

[11] G. Guy and G. Medioni, "Inferring Global Perceptual Contours From Local Features," *Int'l J. Computer Vision*, vol. 20, no. 1/2, pp. 113-133, 1996.

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

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

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

[15] D.G. Lowe, "Three-Dimensional Object Recognition From Single Two-Dimensional Images," *Artificial Intelligence*, vol. 31, pp. 355-395, 1987.

[16] R. Mohan and R. Nevatia, "Segmentation and Description Based on Perceptual Organization," *Proc. Computer Vision and Pattern Recognition*, pp. 333-341, San Diego, 1989.

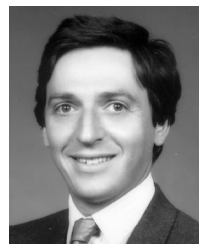
[17] R. Mohan and R. Nevatia, "Using Perceptual Organization to Extract 3-D Structures," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 11, no. 11, pp. 1,121-1,139, Nov. 1989.

[18] T. Poggio and F. Girosi, "A Theory of Networks for Learning," *Science*, vol. 247, pp. 978-982, 1990.

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

[20] S.S. Sinha and B.G. Schunck, "Surface Approximation Using Weighted Splines," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 44-49, Lahaina, Hawaii., June 1991.
[21] D. Terzopoulos, "Regularization of Inverse Visual Problems Involving Discontinuities," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 4, pp. 413-424, 1986.
[22] D. Terzopoulos, "Image Analysis Using Multigrid Relaxation Methods," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 129-139, 1986.
[23] 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.
[24] N. Vaidya and K. Boyer, "Discontinuity Preserving Surface Reconstruction Through Global Optimization," *Proc. Int'l Symp. Computer Vision*, pp. 115-120, 1995.

Gideon Guy earned a BSc degree from the Ben-Gurion University, Israel in 1986, an MSc degree from the Polytechnic institute of Brooklyn in 1990, and a PhD degree from the University of Southern California in 1995, all in electrical engineering. Dr. Guy's interests are in the fields of computer vision and fast computer architectures. He is currently in the employ of Intelligent Computer Solutions, Inc., where he holds the vice president of engineering and technologies position, overseeing all engineering and research activities.



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 (USC), Los Angeles, in 1980 and 1983, respectively. He has been with the University of Southern California 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, and object recognition. He has published 35 journal papers and 80 conference papers in the field. Dr. Medioni is a senior member of the IEEE. He has served on program committees of many major vision conferences. He 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, and general cochair of the IEEE Computer Vision and Pattern Recognition Conference in 1997 in Puerto Rico. Dr. Medioni is associate editor of *IEEE Transactions on Pattern Analysis and Machine Intelligence* and the *Pattern Recognition and Image Analysis* journal.