Short Papers

Implicit Simplicial Models for Adaptive Curve Reconstruction

Gabriel Taubin and Remi Ronfard

Abstract—Parametric deformable models have been extensively and very successfully used for reconstructing free-form curves and surfaces, and for tracking nonrigid deformations, but they require previous knowledge of the topological type of the data, and good initial curve or surface estimates. With deformable models, it is also computationally expensive to check for and to prevent self-intersections while tracking deformations. The *Implicit Simplicial Models* that we introduce in this paper are implicit curves and surfaces defined by piece-wise linear functions. This representation allows for local deformations, control of the topological type, and prevention of self-intersections during deformations. As a first application, we also describe in this paper an algorithm for two-dimensional curve reconstruction from unorganized sets of data points. The topology, the number of connected components, and the geometry of the data are all estimated using an adaptive space subdivision approach. The main four components of the algorithm are topology estimation, curve fitting, adaptive space subdivision, and mesh relaxation.

Index Terms—Curve fitting, topology estimation, shape recovery, geometric modeling.

✦

1 Introduction

The reconstruction of curves and surfaces from unorganized sets of data points is an important problem in computer vision. Curves and surfaces can be represented parametrically or implicitly, and depending on the final application, one representation is more suitable than the other. Since parametric curves and surfaces, such as splines [1], allow for a high degree of local control, they are very good for modeling free-form objects, but the topological type of these curves and surfaces is determined by the topology of the domain, and it is difficult to check for, and to prevent self-intersections. The now popular deformable models [4], [6], [13] are all parametric. There has been some recent work on reconstructing surfaces of unknown topology [3], [8], but no new representation is introduced to control the topology and to prevent self-intersections during deformations.

Arbitrary topology can be achieved with implicit curves and surfaces. The *Implicit Simplicial Models* that we introduce in this paper are polygonal curves and polyhedral surfaces not represented as lists of vertices and planar faces, but defined implicitly by piece-wise linear functions. This representation allows for local deformations, control of the topological type, and prevention of self-intersections during deformations. A piece-wise linear function is determined by a simplicial tessellation of its domain, and by the values of the function at the vertices of the mesh. The function is linear in each one of the domain cells. The usual representation of a polygonal curve or polyhedral surface as a list of vertices and a list of flat faces can be recovered in time proportional to the number of faces. Irregular meshes allow for adaptive reconstruction algorithms, and for hierarchies of curves and surface approximations of different resolutions.

While the *topology* of an implicit simplicial model is determined by the combinatorial structure of the domain mesh, and by the signs of the values of the piece-wise linear function at the vertices of the mesh, the *geometry* is determined by the magnitudes of the values of the piece-wise linear function at the vertices of the mesh. Small scale constant topology deformations can be achieved by changing the magnitudes of the implicit function at the vertices of the mesh keeping the signs fixed. Large scale constant topology deformations are obtained by also deforming the underlying mesh without inverting the orientation of the cells.

This paper builds upon previous work on algebraic curve and surface fitting [9], [10], [11], [12], and some ideas from [7], where an algorithm for adaptively reconstructing piece-wise algebraic curves and surfaces defined on triangular and tetrahedral meshes is described.

While the reconstruction algorithm is applied only to two-dimensional curves here, the representation is valid in any dimension, and the general structure of the reconstruction algorithm can be generalized to higher dimension as well. However, since results were not available at the time this paper was written, we will discuss how to extend the reconstruction algorithm of this paper to surfaces, and how to track deformations, in future reports.

2 Implicit Simplicial Curves

The data structure commonly used to represent a simplicial curve is a pair of lists, a list of vertices, and a list of straight line segments. This representation is good for rendering the curve in time proportional to the number of segments, but it is not very good for checking for, least for imposing, topological or geometric constraints. Even if we start with a valid simplicial curve, deformations can introduce self-intersections, and there is no simple way to check for them. A straightforward check for self-intersections requires time proportional to the square of the number of segments. The main problem with this representation is that the conditions for a set of line segments to define a valid simplicial curve are global.

An implicit simplicial curve is the set of zeros of a piece-wise linear function of two variables, which is defined by a planar triangular mesh and the values of the function at the vertices of the mesh. Thus, we will represent an implicit simplicial curve as a set of three lists $C = \{V, T, F\}$. A list of vertices $V = \{v_1, \ldots, v_{n_V}\}$, a list of triangles $T = \{t_1, \ldots, t_{n_T}\}$, and a list of function values at the vertices of the mesh $F = \{F_1, \ldots, F_{n_V}\}$. Since we will also need later on an explicit representation for the edges of the mesh, we will write $\Sigma = \{V, E, T\}$ for the domain mesh, where $E = \{e_1, \ldots, e_{n_E}\}$ is the list of edges. The piece-wise linear function defining the implicit simplicial curve is

$$f = \sum_{i=1}^{n_V} F_i x_i = FX, \tag{2.1}$$

where F is seen as a row vector, $X = \{x_1, \ldots, x_{n_V}\}^t$, and x_i is the unique piece-wise linear function subordinated to the mesh Σ

• *G. Taubin is with IBM T.J.Watson Research Center, P.O.Box 704, Yorktown Heights, NY 10598. E-mail: taubin@watson.ibm.com.*
• *R. Ronfard is with Institut National de l'Audiovisuel, 94366 Bry sur Marne Cedex, France. He was at the IBM T.J.Watson Research Center when this work was performed. E-mail: remi@ina.fr.*

Manuscript received June 1, 1993; revised May 30, 1995. Recommended for acceptance by Eklundh.
For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number P95153.

which satisfies the following equation

$$x_i(v_j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases}.$$

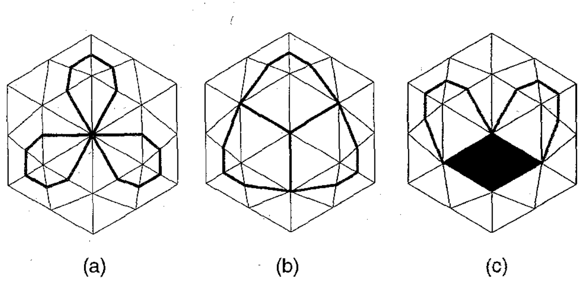

Fig. 1. Singular cases. (a): function is zero at one vertex. (b): Function is zero on three edges. (c): Function is zero on two neighboring triangles.

To prevent singular cases, such as those shown in Fig. 1, we will constrain the values of the piece-wise linear function at the vertices of the mesh to be non-zero. We will also require the domain mesh Σ to be positively oriented. A mesh $\Sigma = \{V, E, T\}$ is positively oriented if all the determinants

$$|V_t| = \begin{vmatrix} 1 & 1 & 1 \\ v_{i,1} & v_{j,1} & v_{k,1} \\ v_{i,2} & v_{j,2} & v_{k,2} \end{vmatrix}$$

associated with the triangles $t = \{v_i, v_j, v_k\}$ of the mesh, are positive, where $v_{i,1}, v_{i,2}$ are the coordinates of the vertex v_i of the mesh. Note that if the order of the vertices is interchanged in a triangle $t = \{v_j, v_i, v_k\}$ the sign of the determinant $|V_t|$ changes.

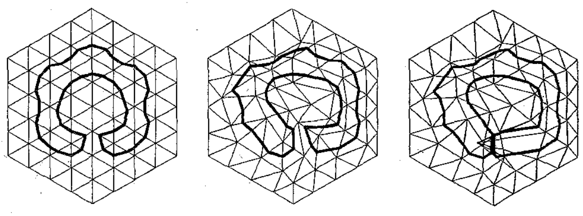

Fig. 2. Topology preserving deformations. (a): Original implicit simplicial curve. (b): Deformed mesh with constant external boundary. (c): Invalid mesh deformation can produce self-intersections.

Topology preserving deformations of a simplicial implicit curve are obtained automatically by deforming the domain mesh maintaining its orientation, *warping* the space around the curve, and eventually changing the magnitudes of the function values at the vertices of the mesh, but not their signs. If a mesh is deformed preserving its orientation, the outside boundary of the mesh might change shape, but preserves its topology. In our reconstruction algorithms we will impose a stronger constraint. We will keep the outside boundary of the mesh constant.

Once the mesh Σ is fixed, the *topology* of an implicit simplicial curve is fully determined by the *signs* of the piece-wise linear function f at the vertices of the mesh, $\{\sigma_1, \ldots, \sigma_{n_V}\}$, $\sigma_i = sign\,(F_i) \in \{-1, 1\}$, while the *geometry* of the curve is determined by the *magnitudes* of the same function values $\{|F_1|, \ldots, |F_{n_V}|\}$.

3 An Algorithm for Curve Reconstruction

In this section we describe an algorithm to reconstruct an implicit simplicial curve from an unorganized set of points in the plane $D = \{p_1, \ldots, p_{n_D}\}$. The only assumption is that the data points belong to, or are close to, a non-singular curve (without self-intersections), and are roghly uniformly distributed along the curve. The topology and the number of connected components are unknown in advance, and estimated by the algorithm. The result is a triangular mesh covering a neighborhood of the data set, and a regular piece-wise linear function represented by its values at the vertices of the mesh. Fig. 3 shows the global structure of the algorithm, and Fig. 4 shows typical examples of curves reconstructed with this algorithm.

```
procedure **FitImplicitSimplicialCurve**
  InitializeMesh
  for level ← 0 to max-level step 1 do
    EstimateTopology
    EstimateGeometry
    if MaximumFittingError < $\epsilon$
      return
    else
      AdaptivelySubdivideMesh
      RelaxMesh
```

Fig. 3. Global structure of the implicit simplicial curve reconstruction algorithm.

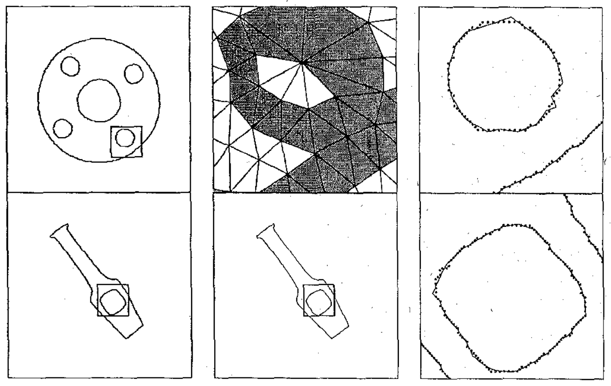

Fig. 4. Multiply connected objects and simplicial curve reconstruction of boundaries, using the algorithm of Section 3—(a): data set; (b): reconstructed implicit simplicial curve; (c): details of a and b.

The algorithm follows a strict top-down approach, minimizing the amount of storage required. A very simple mesh is used at the beginning of the algorithm covering a region containing all the data points. In our experiments, we have used triangulated squares and regular hexagons as initial meshes, and in general, the hexagonal geometries produce better results. However, the initial mesh should be tailored to the application. Once a mesh is fixed, an implicit simplicial curve subordinated to the current mesh is reconstructed from the data by a least squares fitting algorithm, by first estimating the topology, and then the geometry of the curve. After the curve fitting step, the maximum fitting error within each triangle is measured. If all the triangles meet the prespecified tolerance, the algorithm finishes. Otherwise, the triangles where the tolerance is not met, are subdivided, along with a few other that are necessary to maintain a valid mesh. The mesh is then relaxed to prevent the vertices of the subdivided mesh to be too close to the data, and to improve the aspect ratio of the triangles. This step is essential for the success of the algorithm. Once this new mesh is fixed, the loop is traversed again, until the tolerance test is satisfied by all triangles.

Now we proceed to describe in detail the main building blocks of the curve reconstruction algorithm: implicit curve fitting, topology estimation, testing, adaptive subdivision, and mesh relaxation.

3.1 Implicit Curve Fitting

We formulate this step of the algorithm building upon previous work on algebraic curve and surface fitting [9], [10], [11], [12]. Given a finite set of two dimensional data points $D = \{p_1, \ldots, p_{n_D}\}$, we cast the problem of fitting an implicit curve $Z(f) = \{p: f(p) = 0\}$ to the data set D as globally minimizing the mean square distance from the data points to the curve $Z(f)$, as a function of the vector of parameters F, the values of the piece-wise linear function on the vertices of the mesh. For a piece-wise linear function, the mean square distance has the following explicit expression

$$\Delta_1(F) = \sum_{t\in T} \frac{n_{D_t}}{n_D} \sum_{p\in D_t} \frac{f_t(p)^2}{\|\nabla f_t(p)\|^2} = \sum_{t\in T} \frac{n_{D_t}}{n_D} \frac{F_t M_t F_t^t}{F_t N_t F_t^t}, \tag{3.1}$$

where D_t is the subset of data points that belong to the triangle $t = (v_i, v_j, v_k)$, n_{D_t} is the number of points in D_t, $f_t(p) = F_i x_i(p) + F_j x_j(p) + F_k x_k(p)$ is the restriction of $f(p)$ to the triangle t, $F_t = (F_i, F_j, F_k)$, $X_t = (x_i, x_j, x_k)$,

$$M_t = \frac{1}{n_{D_t}} \sum_{p\in D_t} \left[X_t(p) X_t(p)^t\right], N_t = \frac{1}{n_{D_t}} \sum_{p\in D_t} \left[DX_t(p) DX_t(p)^t\right], \tag{3.2}$$

and DX_t is the Jacobian of X_t

In fact, since X_t is a linear function, its Jacobian is constant, and the matrix N_t is only a function of the vertices of the triangle, and not a function of the data points inside it.

3.2 Topology Estimation

In the absence of prior knowledge about the solution, it is difficult to minimize (3.1), because the system to be solved becomes singular when a nodal value F_i approaches zero. On the other hand, if we can estimate the signs of the nodal values, i.e., the topology of the solution, we can minimize (3.1) locally, with a program such as LBFGS [5], which is designed for large unconstrained nonlinear minimization.

The piece-wise linear function determined by the coefficients F_i constitutes an approximate *inside-outside* function for the data (up to a global sign inversion). When an inside-outside function is directly available from the data, the topology estimation step is therefore not necessary. In all other cases, we can still estimate the inside-outside function, with combinatorial optimization methods. We do this by independently fitting straight lines in all non-empty triangles, and *counting sign changes*. More specifically, for each triangle $t = \{v_i, v_j, v_k\}$, i.e., such that the set D_t is not empty, we fit a straight line to the data set D_t in the least squares sense by minimizing the local mean square error

$$\frac{L_t M_t L_t^t}{L_t N_t L_t^t},$$

where M_t and N_t are the matrices of (3.2), obtaining its minimum value, the *local error of fit* ϵ_t, and its minimizer $L_t = (L_{t,i}, L_{t,j}, L_{t,k})$. This is done for each triangle independently of the data in other triangles. This can be done in closed form and involves solving a 2×2 eigenvalue problem. In general, the three components of L_t are nonzero. Let us denote by $\sigma_{t,i}, \sigma_{t,j}, \sigma_{t,k}$ the signs of $L_{t,i}, L_{t,j}, L_{t,k}$, i.e., $\sigma_{t,i}$ is equal to 1 or −1 depending on whether $L_{t,i}$ is positive or negative.

A good estimate for the signs of the global coefficients F_i can then be obtained by minimizing a sum over all triangles:

$$\Delta_2(F) = \frac{1}{n_T} \sum_{t\in T, t=\{v_i,v_j,v_k\}} \left(\sigma_{t,i}\sigma_{t,j}F_iF_j + \sigma_{t,i}\sigma_{t,k}F_iF_k + \sigma_{t,j}\sigma_{t,k}F_jF_k\right)\left(\frac{\epsilon_{\max} - \epsilon_t}{\epsilon_{\max} - \epsilon_{\min}}\right) \tag{3.3}$$

constraining the function values at the vertices of the mesh to be either −1 or 1

$$F_1, \ldots, F_{n_V} \in \{-1, 1\}. \tag{3.4}$$

Expression (3.3) involves a measure of the *goodness-of-fit* for each triangle, where $\epsilon_{\max} = \max\{\epsilon_{t_1}, \ldots, \epsilon_{t_{n_T}}\}$, and a similar definition for $\epsilon_{\min}$. Thus, the global coefficients F_i change signs only when there is enough evidence from the local fits in their neighborhood. Of course, we also have to define a goodness of fit and a fitting vector L_t for each empty triangle, because otherwise the problem could be underconstrained. For an empty triangle t we set $L_t = \{1, 1, 1\}$ with a high confidence value $e_t = e_{empty}$. By rearranging terms, we obtain

$$\Delta_2(F) = \sum_{e\in E} H_e F_i F_j \tag{3.5}$$

with the sum ranging over all the edges $e = \{v_i, v_j\}$ of the triangulation. The edge weight H_e corresponding to the edge $e=\{v_i,v_j\}$ is easily obtained from (3.3)

$$H_e = -\sum_{t:e\in t} \sigma_{t,i}\sigma_{t,j}\left(\frac{\epsilon_{\max} - \epsilon_t}{\epsilon_{\max} - \epsilon_{\min}}\right) \tag{3.6}$$

with the sum extended over all the triangles that contain e as an edge, The minimization of the quadratic expression (3.5) in {−1, 1} is exactly the Ising model, for which simulated annealing schemes are well documented. We have found that simulated annealing based on (3.5) gives good results. This approximation is also faster, and more robust, than the more obvious choice of using (3.1) directly at this stage. Some results of the topology estimation step are presented in Figs. 5 and 6.

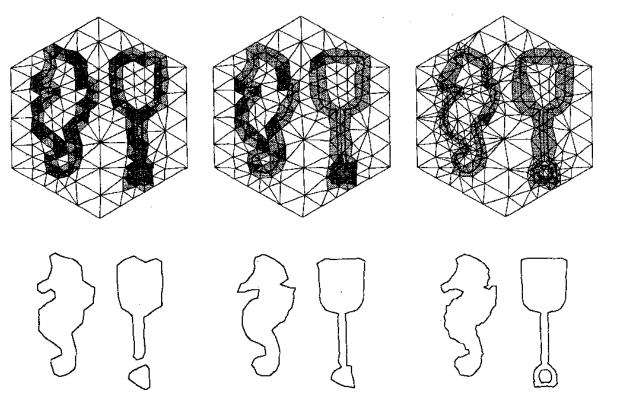
Fig. 5. Example of topology estimation step.

3.3 Testing

Testing the quality of the fit is necessary to determine when the algorithm should stop, and otherwise, which triangles need to be subdivided. After the minimization of (3.1) for the current mesh, for each triangle t we compute

$$\delta_t = \max_{p\in D_t} \frac{f_t(p)^2}{\|\nabla f_t(p)\|^2}.$$

If $\delta_t > \delta$ we mark the triangle for subdivision. If no triangle is marked, the algorithm stops. Otherwise we continue. In practice, this test is generally sufficient, although problems occur when the number of data points inside a triangle becomes too small, usually around high curvature regions, or in cases of sparse or very noisy data points.

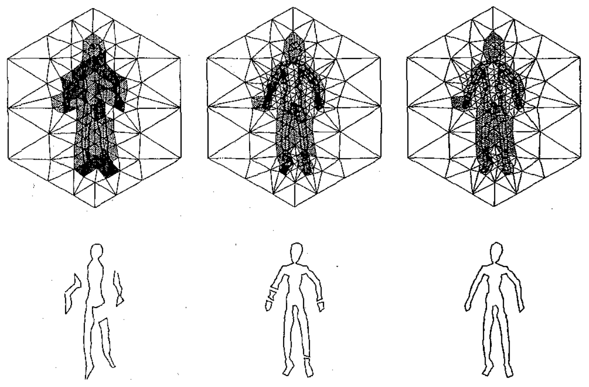

Fig. 6. Example of topology estimation step.

3.4 Adaptive Subdivision

At this point we have a valid triangular mesh with some triangles marked for subdivision. It turns out that to maintain a valid mesh, other unmarked triangles may have to be subdivided as well. A simple method to automate the process involves three steps [2], [14]. In the first step the vertices of each triangle marked for subdivision are marked. In the second step a new vertex is created at the midpoint of each edge which has the two vertices marked. In the third step, illustrated in Fig. 7, the triangles are subdivided according to how many vertices are marked. A triangle with the three vertices marked is subdivided into four triangles, a triangle with two vertices marked is subdivided into two triangles, and triangles with one or no marked vertices are not subdivided.

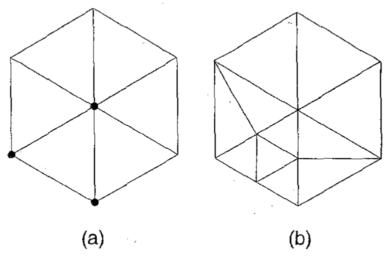

Fig. 7. Subdivision rules. Marked vertices are represented with a circle. (a): Mesh with some vertices marked. (b): Mesh a after subdivision.

3.5 Mesh Relaxation

Since the desired final result is a regular implicit simplicial curve, we cannot allow the function to be close to zero at any vertex of the triangulation. If a vertex is allowed to get close to the data points, the function value at that vertex will tend to be small with respect to the values a the other vertices, or even zero. Contrary to what is done in other related algorithms based on triangular meshes, where the mesh is relaxed by pulling the vertices close to the data points [7], our mesh relaxation process *pushes* the vertices *away from the data*. It is not only that we want the vertices to be far away from the data, but we also want the data inside each triangle to be well approximated by a straight line connecting the midpoints of two edges. At least this is approximately what happens when the algorithm stops, but we would like to try to impose this condition at each level of subdivision. So, the data has to pull the vertices away, but at the same time, for each triangle the distance from the three vertices to the data must be as equal as possible. Also, the mesh has to remain a valid mesh, and the triangles have to remain as equilateral as possible, because otherwise, the local nonlinear minimization algorithm gets plagued with all sort of numerical problems.

We have decided to base our mesh relaxation algorithm on an energy minimization scheme, and instead of solving ordinary differential equations, we perform an approximate minimization based on gradient descent.

The mesh energy $U = \kappa_D U_D + \kappa_E U_E$ has two components, the *data energy* U_D, and the *edge energy* U_E. The two constants κ_D and κ_E must be positive. In our current implementation $\kappa_E = 0.25$ and $\kappa_D = 1.0$.

The data energy pushes the vertices as far away and equidistantly as possible from data points in each triangle. The edge energy pulls vertices together and tends to make equilateral triangles, which regularizes the mesh relaxation process. In addition, we constrain the boundary of the mesh to remain constant. This can be achieved by fixing the boundary vertices in their initial positions, and allowing the vertices laying on the boundary edges to move only along the edges they belong to. All of this can be done with linear constraints on some of the vertices. If a vertex v of the mesh belongs to a boundary edge e, then v must satisfy the linear equation of the line containing the edge $C_e(v) = 0$. Since a boundary vertex belongs to two non-parallel edges, making it satisfy the two constraints is equivalent to keeping it fixed.

The data energy is defined as follows

$$U_D = \rho^2 \sum_{e\in E} \left(\phi(v_i) - \phi(v_j)\right)^2, \tag{3.7}$$

with the inner sum extended over all the edges $e = \{v_i, v_j\}$, and where

$$\phi(v_i) = \frac{1}{n_D} \sum_{t:v_i\in t} \sum_{p_j\in D_t} \frac{1}{\|v_i - p_j\|}, \tag{3.8}$$

with the sum extended over all the triangles that contain v_i as a vertex, and all the data points inside these triangles. The constant ρ is the diameter of the mesh, and the reason to include the factor ρ^2 here is to make the data energy scale invariant. Scale invariance is important only to be able to define the mesh energy as a linear combination of the data energy and the edge energy, independently of the scale of the problem.

The edge energy is defined as follows

$$U_E = \frac{1}{\rho^2} \sum_{e\in E} \nu_i \nu_j \|v_i - v_j\|^2, \tag{3.9}$$

where $e = \{v_i, v_j\}$, and $\nu_1, \ldots, \nu_{n_V}$ are positive constants defined as follows. The *degree* of a vertex is the number of edges incident to the vertex, or equivalently, the number of vertices connected to the former one through an edge of the triangulation. The *mean degree* of a mesh is the mean value of the degrees of its vertices. In our implementation we have defined the constant ν_i as the ratio of the degree of the vertex v_i over the mean degree of the mesh, but other values provide similar results. For example, making $\nu_i = 1$ for all i is also a good choice. The factor ρ^2 is included in the denominator to make the edge energy scale invariant.

Special care should be taken in choosing the energy minimization algorithm, because since each time the vertices are moved, the data set has to be repartitioned, evaluating the energy function is potentially expensive. In principle, after a mesh deformation each point must be tested against each triangle for membership, but for small deformations each data point most likely will remain in the same triangle, or will move to one of the three neighbors.

4 Conclusions

We have introduced implicit simplicial models as a new representation for piece-wise linear curves and surfaces. We have shown that this new representation allows for a complete and efficient control of the topology of the curve or surface, and has most of the good properties of more traditional deformable models, and algebraic curves and surfaces. Implicit simplicial models can be used to model free-form curves and surfaces, but at the same time they provide an inside-outside function defined in a large neighborhood of the curve or surface. This inside-outside function can be constructed as an approximation to the distance from an arbitrary point to the curve or surface. At the same time, implicit simplicial curves and surfaces have explicit local parameterizations, which are good for other purposes. As a first application, we have described a two dimensional curve reconstruction algorithm from unorganized data sets which can be extended with almost no modification to an algorithm for surface reconstruction. We believe that a number of graphics and vision problems can be solved either more robustly, more generally, or both using implicial simplicial models, as for example surface reconstruction, tracking of surface deformations and adaptive isosurface construction, to mention just a few applications that we intend to demonstrate in future reports.

References

[1] G.E. Farin, *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* New York: Academic Press, 1988.

[2] M. Hall and J. Warren, "Adaptive polygonalization of implicitly defined surfaces," *IEEE Computer Graphics and Applications*, pp. 33–42, Nov. 1990.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics*, pp. 71–78, July 1992. (*Proc. SIGGRAPH '92.*)

[4] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int'l J. Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[5] D. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. B45, pp. 503–528, 1989.

[6] D. Metaxas and D. Terzopoulos, "Dynamic deformation of solid primitives with constraints," *Computer Graphics*, vol. 26, no. 2, pp. 309–312, July 1992.

[7] D. Moore and J. Warren, "Approximation of dense scattered data using algebraic surfaces," Technical Report Rice COMP TR90-135, Department of Computer Science, Rice University, Houston, Oct. 1990.

[8] R.S. Szeliski and D. Tonnesen, "Surface modeling with oriented particle systems," *Computer Graphics*, pp. 185–194, July 1992. (*Proc. SIGGRAPH '92.*)

[9] G. Taubin, "Nonplanar curve and surface estimation in 3-space," *Proc. IEEE Conf. Robotics and Automation*, pp. 644–645, May 1988.

[10] G. Taubin, "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, pp. 1,115–1,138, Nov. 1991.

[11] G. Taubin, "An improved algorithm for algebraic curve and surface fitting," *Proc. Fourth Int'l Conf. Computer Vision*, pp. 658–665, Berlin, Germany, May 1993.

[12] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman, "Parameterized families of polynomials for bounded algebraic curve and surface fitting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 3, pp. 287–303, Mar. 1994.

[13] D. Terzopoulos and K. Fleischer, "Deformable models," *The Visual Computer*, vol. 4, pp. 306–311, 1988.

[14] G.L. Tindle, *The Mathematics of Surfaces II*, pp. 387–394. Oxford, England: Clarendon Press, 1987.