Curvature and Continuity Control in Particle-Based Surface Models

Richard Szeliski[1], David Tonnesen[2], and Demetri Terzopoulos[2]

[1]Digital Equipment Corporation, Cambridge Research Lab, One Kendall Square, Bldg. 700, Cambridge, MA 02139
[2]Department of Computer Science, University of Toronto, Toronto, ON, M5S 1A4

Abstract

This paper develops techniques to locally control curvature and continuity in particle-based surface models. Such models are a generalization of traditional spline surfaces built out of triangular patches. Traditional splines require the topology of the triangular mesh to be specified ahead of time. In contrast, particle-based surface models compute the topology dynamically as a function of the relative node positions, and can add or delete nodes as required. Such models are particularly important in computer vision and other inverse problems, where the topology of the surface being reconstructed is usually not known *a priori*. We develop techniques for both locally controlling the curvature of the surface (through additional state at each node), and for adapting the triangulation to surface curvature (by concentrating more particles in areas of high curvature). We show how the same ideas can also be applied to 3-D curves, which results in a flexible version of traditional dynamic contours (snakes).

1. Introduction

Flexible 3D surface representations are an essential component of 3D computational vision, enabling the estimation of geometric shape from various types of visual data, including range and surface normal measurements. These surfaces can be used as an intermediate representation for object recognition, to guide robotics tasks such as grasping, to segment three-dimensional volumes (e.g., in medical applications), and to integrate different visual modalities such as stereo and shading.

Existing surface representations have limitations—viewer-centered representations [3, 17, 21, 27] do not describe non-visible portions of object surfaces, while the object-centered representations [1, 16, 23] often make strong assumptions about object topology. Unknown object topology poses difficult challenges to surface modeling and reconstruction that have so far been largely ignored in the vision literature. Unfortunately, vision systems that must derive quantitative models of complex real-world objects from multiple views cannot avoid the issue of unpredictable topological structure. Unknown topology is also an important concern in the related fields of biomedical and geological imaging where there is a need to analyze three dimensional arrays of volumetric density or reflectivity data.

One way to cope with unknown topological structure is to use a "patchwork" representation that describes the surface only locally in terms of planar, quadric, or cubic patches [15]. The lack of a global, continuous representation makes this approach cumbersome for surface analysis tasks such as area, curvature, and enclosed volume computations. More serious difficulties arise in the dynamic analysis of objects, including the incremental reconstruction of surfaces from sequential views around objects, or the reconstruction, tracking, and motion estimation of dynamic nonrigid objects such as a beating heart. A globally consistent surface model can provide powerful constraints for solving these dynamic estimation problems.

In previous work, we developed a flexible surface model based on *oriented particles* which can model surfaces of arbitrary (and unknown) topology [19] and fit these surfaces to 3D range or volumetric data [20]. Our approach retains the topological flexibility of the local patch methods, while constructing globally coherent surface models that can evolve consistently with time-varying data. Our representation is three-dimensional and viewpoint invariant. It is also non-parametric, i.e., it does not represent the surface as a function of a two-dimensional parameter space. Instead, all information is represented in local differential geometric quantities (orientations, curvatures) which are used to form continuous surfaces.

In our previous papers [19, 20], we concentrated on the discrete particle-based nature of our surface model. In this paper, we first focus on continuous models of curves and surfaces, and then show how to implement discrete oriented particle systems that approximate these models. Both the continuous and discrete models are based on collections of oriented trihedral coordinate frames which represent position and local differential geometric information such as orientation and curvature. For the discrete model, we design special interaction potentials which favor local arrangements of particles that are consistent with continuous surfaces. We also use potential functions borrowed from molecular dynamics and computer graphics [24, 25] to control particle spacing. For the continuous model, we develop cubic interpolating splines (Bézier curves and triangles) which "skin" the particles to form continuous geometric models. We show how 3D curves can also be modeled using a similar local geometric representation by adding additional potentials to our oriented particle system. We also develop a method for concentrating more particles in areas of high curvature, thus improving the accuracy of the representation.

The remainder of the paper is organized as follows. The next section demonstrates the flexibility of our dynamic particle approach with some examples of interactive surface modeling and surface reconstruction. Section 3 briefly reviews the differential geometry of curves and surfaces. Section 4 formulates our oriented particles and their associated interaction potentials which favor continuous smooth curves and surfaces. It also introduces new potentials for locally controlling curvature and continuity. Section 5 discusses the numerical time integration of the equations of motion of the particle system and related complexity issues. Section 6 presents an algorithm for triangulating particles into globally coherent surfaces and for interpolating smoothly within and between triangles using Bézier curves and triangles. We close with a discussion of alternative surface reconstruction techniques and directions for future work.

2. Modeling and Reconstructing Surfaces with Particles

Our approach to surface modeling and reconstruction has two components. The first is a dynamic particle system which discovers topological and geometric surface structure implicit in the data. The second component is an efficient triangulation and interpolation scheme which connects the particles into a continuous global surface model that is consistent with the particle structure. The evolving global model supports the automatic extension of existing surfaces with few restrictions on connectivity, the joining of surfaces to form larger continuous surfaces, and the splitting of surfaces along arbitrary discontinuities as they are detected.

To demonstrate the some of the power of our approach, Figure 1 demonstrates an interactive modeling system in which a flat, dynamic particle surface is deformed into a complex surface with a looped handle. Figure 2 shows another example where we start with a spherical dynamic particle surface, and by pushing in the two ends, form it into a torus. New particles are created on the surface due to stretching during the formation process, and some old particles are deleted when they become trapped between the two spherical shaping tools.

Another important application of our oriented particle systems is the interpolation and extrapolation of sparse 3-D data. This is a particularly difficult problem when the topology or approximate shape of the surface to be fitted is unknown. Our particle system solves this problem by spawning particles between sample points to interpolate a continuous surface (Figure 4). When the surface being reconstructed has holes or gaps, we can control the size of gaps that are filled in by modifying a threshold.

Dynamic particle surface models may be used to help segment structures in 3-D volumetric data such as CT, MRI, or other 3-D medical imagery. To perform this segmentation, we first apply a 3-D edge operator to the data and use the edges to initialize and attract particles. Figure 3a–c shows slices from a CT scan of a plastic "phantom" vertera model (decimated to $120 \times 128 \times 52$ resolution). Figure 3d shows the reconstructed 3-D model. This smooth, triangulated model contains 6650 particles and 13829 triangles, and was created by seeding a single particle and extending the surface along high edge values until a closed surface was obtained. Figure 3d shows a Gouraud shaded rendition of the reconstructed surface.

3. Differential Geometry of Curves and Surfaces

The basic approach to modeling curves and surfaces in this paper is as collections (chains or meshes) of local geometric descriptors. The study of such descriptors is the domain of differential geometry [9, 7].

The local shape of a 3-D space curve $\mathbf{x}(t)$ can most easily be described with respect to its *Frenet frame* (Figure 5a), which consists of the *tangent vector* $\mathbf{t}$, the *main normal vector* $\mathbf{m}$, and the *binormal vector* $\mathbf{b}$.[1] The formulas for these three vectors are

$$\mathbf{t} = \mathcal{N}(\dot{\mathbf{x}}), \;\; \mathbf{b} = \mathcal{N}(\dot{\mathbf{x}} \times \ddot{\mathbf{x}}), \;\; \mathbf{m} = \mathbf{b} \times \mathbf{t}, \tag{1}$$

where dots denote derivatives with respect to t, $\mathcal{N}(\cdot)$ normalizes a vector, and "$\times$" is the cross product. The variation of the Frenet frame as we move along the curve describes the local shape of the curve. If we re-parameterize the curve so that it is a function of arclength $\mathbf{x}(s)$ [7], the *Frenet-Serret* equations describe this evolution,

$$\begin{bmatrix} \mathbf{t}' \\ \mathbf{m}' \\ \mathbf{b}' \end{bmatrix} = \begin{bmatrix} 0 & \kappa & 0 \\ -\kappa & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{m} \\ \mathbf{b} \end{bmatrix}, \tag{2}$$

[1]We use $\mathbf{m}$ for the main normal vector to avoid confusion with the surface normal $\mathbf{n}$.

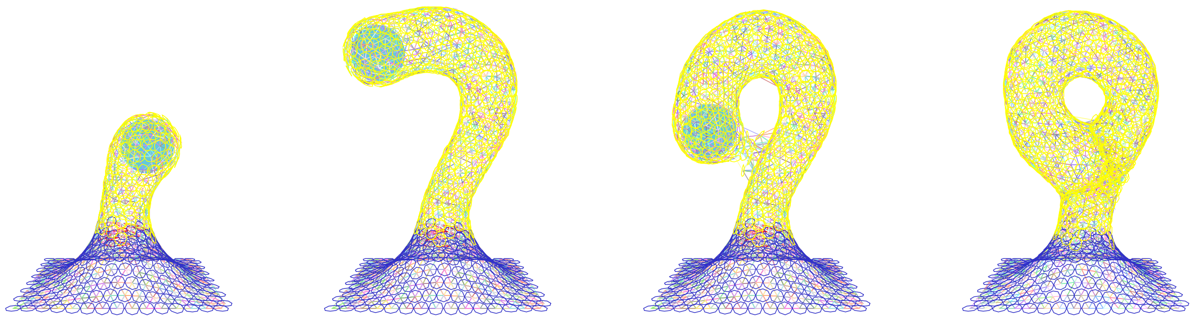


Figure 1: Forming a complex object. The initial surface is deformed upwards and then looped around. The new topology (a handle) is created automatically.

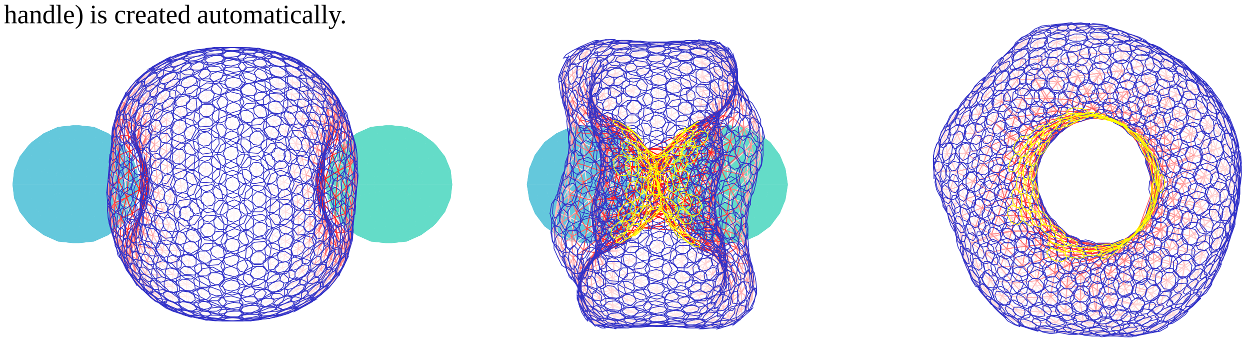


Figure 2: Deformation from sphere to torus using two spherical shaping tools. The final view is from the side, showing the toroidal shape.

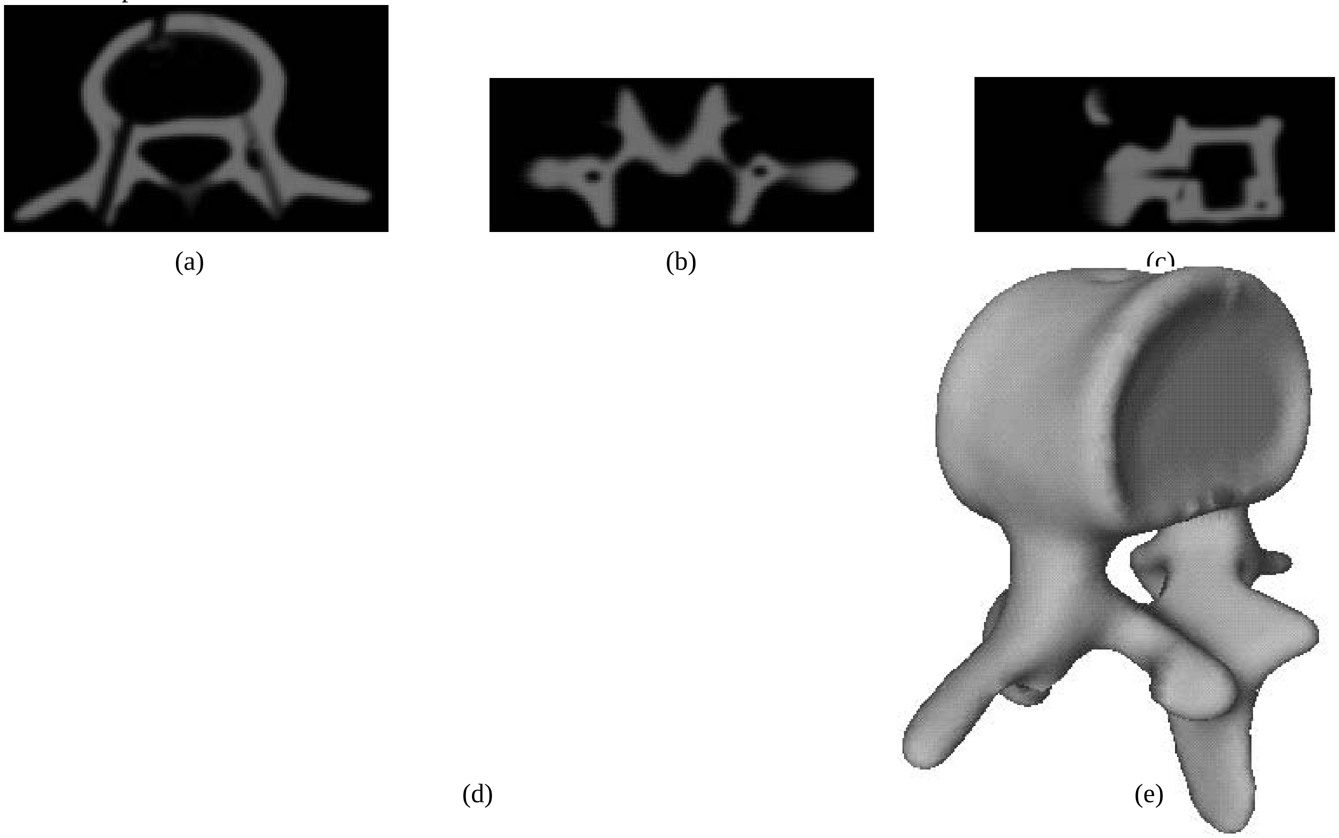


Figure 3: 3-D Reconstruction of a vertebra from 120 × 128 × 52 CT volume data: (a) xy slice, (b) xz slice, (c) yz slice, (d) reconstructed 3-D surface model with triangulated particles, (e) shaded surface.

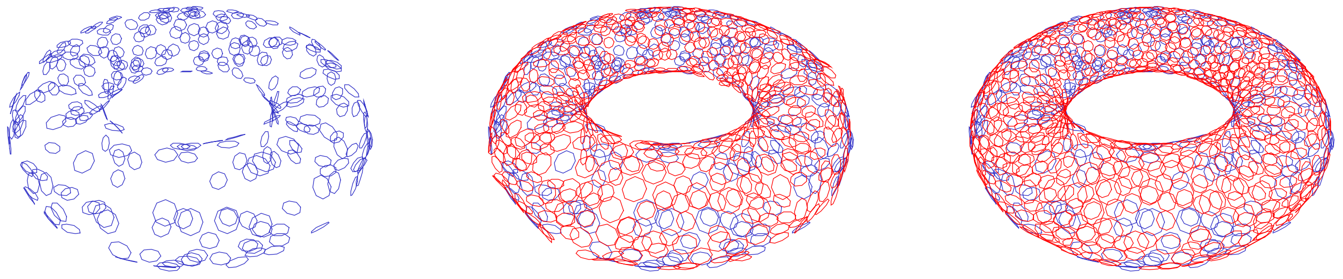


Figure 4: Surface interpolation through a collection of 3-D points. The surface extends outward from the seed points until it fills in the gaps and forms a complete surface.

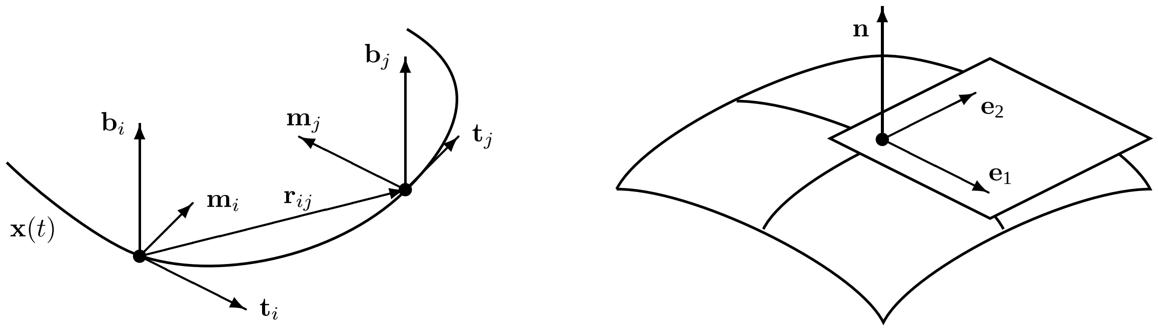


Figure 5: (a) Local Frenet frame for a curve; (b) the principal frame and the tangent plane for a surface patch.

where the primes denote differentiation with respect to arc length s. The curvature and torsion functions, $\kappa(s)$ and $\tau(s)$, are *intrinsic* descriptions of the curve, i.e., they are independent of the choice of coordinate frame. Together with a starting position $\mathbf{x}(0)$ and an initial Frenet frame, they completely determine the shape of a curve. Given a set of Frenet frames and curvatures, we can check the consistency of adjacent frames (Section 4) using the local Taylor series expansion of the curve equation $\mathbf{x}(s)$, i.e., the *Frenet approximation* [9],

$$\mathbf{x}(s) = \mathbf{x}_0 + s\mathbf{t}_0 + \tfrac{1}{2}\kappa s^2\mathbf{m}_0 + \tfrac{1}{6}\kappa\tau s^3\mathbf{b}_0 + \cdots. \tag{3}$$

Just as with a curve segment, the local shape of a 3-D surface $\mathbf{x}(u,v)$ can most easily be described with respect to a particular reference frame, namely the *principal frame* (Figure 5b), which consists of the *normal vector* $\mathbf{n}$, and the two *principal direction vectors* $\mathbf{e}_1$ and $\mathbf{e}_2$. The normal vector $\mathbf{n}$ is the normal to the tangent plane at the origin of the frame, and can be computed from

$$\mathbf{n} = \mathcal{N}(\mathbf{x}_u \times \mathbf{x}_v) \tag{4}$$

where $\mathbf{x}_u$ and $\mathbf{x}_v$ are the partial derivatives of $\mathbf{x}$ with respect to u and v. The normal curvature is the curvature κ_0 of a *normal section*, i.e., the intersection of the surface with a plane containing the normal vector $\mathbf{n}$ (see [7] for equations for κ_0). The principal directions $\mathbf{e}_1$ and $\mathbf{e}_2$ are the directions in the tangent plane of the *normal curvature* extrema κ_1 and κ_2.

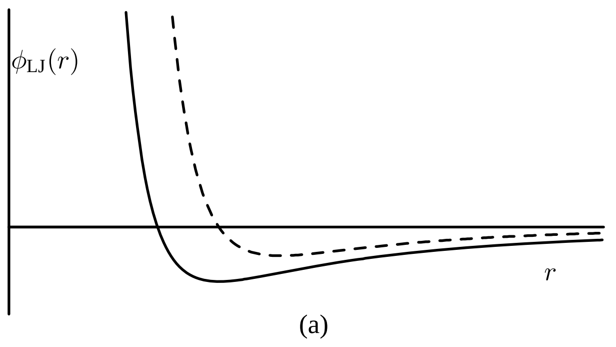
When we align local parameter directions (x,y) with the principal directions $(\mathbf{e}_1,\mathbf{e}_2)$, the *Gauss-Weingarten equations* describe the local motion of the principal frame,

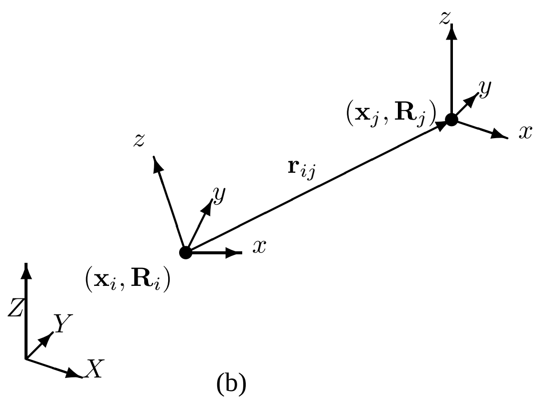
$$\frac{\partial}{\partial x}\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\\ \mathbf{n}\end{bmatrix} = \begin{bmatrix}0 & g_1 & \kappa_1\\ -g_1 & 0 & 0\\ -\kappa_1 & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\\ \mathbf{n}\end{bmatrix}, \quad \frac{\partial}{\partial y}\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\\ \mathbf{n}\end{bmatrix} = \begin{bmatrix}0 & g_2 & 0\\ -g_2 & 0 & \kappa_2\\ 0 & -\kappa_2 & 0\end{bmatrix}\begin{bmatrix}\mathbf{e}_1\\ \mathbf{e}_2\\ \mathbf{n}\end{bmatrix}, \tag{5}$$

where g_1 and g_2 are called the *geodesic curvatures* [9]. We can also write the local shape of the surface as

$$z(x,y) = \tfrac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2) + \cdots, \tag{6}$$

where $z(x,y)$ is the displacement of the surface along the normal direction (the *Monge patch* [9]).





z
y
$(\mathbf{x}_j, \mathbf{R}_j)$
x
z
$\mathbf{r}_{ij}$
y
x
$(\mathbf{x}_i, \mathbf{R}_i)$
Z
Y
X
(b)

Figure 6: (a) Lennard-Jones type function: the solid line shows the potential function $\phi_{\mathrm{LJ}}(r)$, and the dashed line shows the force function $f(r) = -\frac{d}{dr}\phi_{\mathrm{LJ}}(r)$; (b) two interacting oriented particles: the interparticle distance $\mathbf{r}_{ij}$ is computed from the global coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$ of particles i and j.

4. Oriented Particle System

Collections of Frenet frames and principal frames can be used to model continuous curves and surfaces. To be useful, these must be augmented with potential functions that enforce constraints such as consistency between local frames and smoothness. In this section, we describe oriented particles, which are our implementation of these frames, and devise appropriate interaction potentials. We begin with some background material on particle systems, and then develop potential functions for enforcing smoothness, for local curvature control, and for estimating curvature from the data. Section 6 describes how these frames can be interpolated to form continuous curves and surfaces.

4.1. Particle Systems

Ideas from molecular dynamics [8] have been used to develop computer graphics models of fluids and deformable solids using collections of interacting particles [24, 25]. In these models, long-range attraction forces and short-range repulsion forces control the dynamics of the system. Typically, these forces are derived from an intermolecular potential function such as the Lennard-Jones function

$$\phi_{\mathrm{LJ}}(\mathbf{r}_{ij}) = A\|\mathbf{r}_{ij}\|^{-n} - B\|\mathbf{r}_{ij}\|^{-m} \tag{7}$$

(Figure 6), where $r = \|\mathbf{r}_{ij}\| = \|\mathbf{p}_j - \mathbf{p}_i\|$ is the distance between molecules i at $\mathbf{p}_i$ and j at $\mathbf{p}_j$ and m, n, A, B are constants (we use the default values $A = B = 1.0$, $m = 1$, $n = 3$). When external forces are insignificant compared to internal forces, particles will bond together into closely packed structures to minimize their total energy, thereby behaving like solids. As internal forces decrease, the behavior resembles that of viscous fluids.

4.2. Oriented Surface Particles

To model surfaces, we use *oriented particles* with additional interaction potentials. An oriented particle has a position and a local coordinate frame for a total of six degrees of freedom, and represents the local principle frame for the surface, i.e., it defines both a normal vector ($\mathbf{n}_i = z$ in Figure 6b) and a local tangent plane to the surface (defined by the local x and y vectors). More formally, we write the state of each particle as $(\mathbf{p}_i, \mathbf{R}_i)$, where $\mathbf{p}_i$ is the particle's position, $\mathbf{R}_i$ is a 3×3 rotation matrix which defines the orientation of its local coordinate frame relative to the global frame (X, Y, Z), and the third column of $\mathbf{R}_i$ is the local normal vector $\mathbf{n}_i$. Using the differential geometric relationships introduced in Section 3, we devise some additional potential functions that encourage oriented particles to group themselves into surface-like arrangements.

The first potential is based on the squared error in the local surface approximation (6) to encourage formations that are consistent with continuous Monge patches

$$\phi_{\mathrm{P}}(\mathbf{n}_i, \mathbf{e}_{1i}, \mathbf{e}_{2i}, \mathbf{r}_{ij}) = [z - \tfrac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2)]^2 \psi(\|\mathbf{r}_{ij}\|), \tag{8}$$

where $(x, y, z) = \mathbf{R}_i^{-1}\mathbf{r}_{ij} = \mathbf{R}_i^{-1}(\mathbf{p}_j - \mathbf{p}_i)$ gives the coordinates of particle j in the frame of particle i. An additional potential based on the Gauss-Weingarten equation (5) controls the relationship between nearby frame orientations:

$$\phi_{\mathrm{N}}(\mathbf{n}_i, \mathbf{e}_{1i}, \mathbf{e}_{2i}, \mathbf{n}_j, \mathbf{r}_{ij}) = \|\mathbf{n}_i - \kappa_1 x \mathbf{e}_{1i} - \kappa_2 y \mathbf{e}_{2i} - \mathbf{n}_j\|^2 \psi(\|\mathbf{r}_{ij}\|). \tag{9}$$

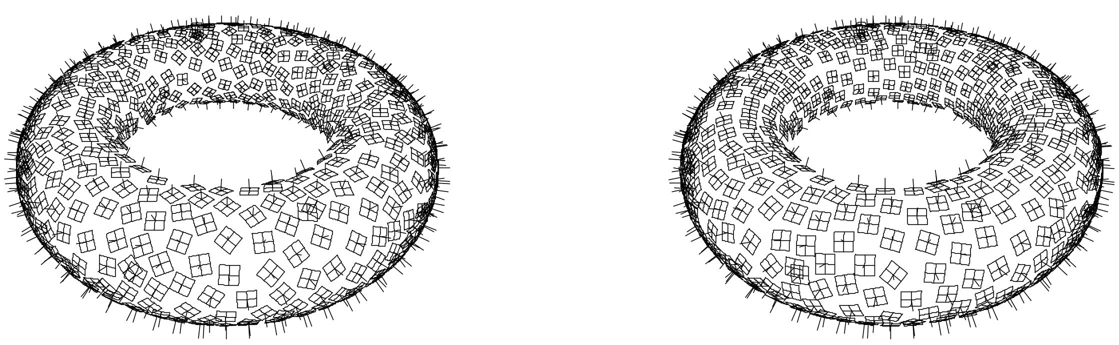


Figure 7: The spin torque potential $\phi_{\rm S}$ forces the local coordinate frames to align with the minimum and maximum curvatures of the surface. The left and right images are before and after the addition of $\phi_{\rm S}$. The quadratic patches also indicate the amount of estimated local curvature.

Note that the above potentials are more general than the ones we defined in [19, 20] whose rest (minimum energy) configurations are planes. In particular, $\phi_{\rm P}$ and $\phi_{\rm N}$ reduce to our previous co-planarity and co-normality constraints, respectively, for $\kappa_1 = \kappa_2 = 0$.

To control the deformation characteristics of our deformable surface, we use a weighted sum of the above potential energies

$$E_{ij} = \alpha_{\rm LJ}\phi_{\rm LJ}(\|\mathbf{r}_{ij}\|) + \alpha_{\rm P}\phi_{\rm P}(\mathbf{n}_i, \mathbf{r}_{ij}) + \alpha_{\rm N}\phi_{\rm N}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) \tag{10}$$

The first term controls the average inter-particle spacing, the next two terms control the surface's resistance to bending and the last controls the surface's tendency towards uniform local curvature. We use the default values $\alpha_{\rm LJ} = 2.0$, $\alpha_{\rm P} = 1.7$, $\alpha_{\rm N} = 1.0$. The total internal energy of the particle system $E_{\rm int}$ is computed by summing the inter-particle energies over all interacting neighbors (Section 5).

4.3. Estimating Local Surface Curvature

In data fitting applications, the local curvature at each particle may have to be estimated from the particle positions and orientations. This can be done by adding a potential function that induces a torque around the local z axis

$$\phi_{\rm S}(x, y, z) = x^2 z\, \psi(\|\mathbf{r}_{ij}\|). \tag{11}$$

This potential forces the x and y axes to align themselves in the directions of minimum and maximum curvature. In order not to disturb the original dynamics of the surface, the above potential is used only to compute a torque around the local z axis.

Adding these curvature-based torques to our particle system results in a covering of the surface with local coordinate frames that indicate the principal directions of the surface at each point (Figure 7). The minimum and maximum curvature values can be computed by minimizing (8) with respect to κ_1 and κ_2, i.e., adding forces on local curvature estimates. The resulting system of oriented particles resembles the collection of interacting Darboux frames used by Sander and Zucker [15].

4.4. Curvature Dependent Particle Spacing

Once we have a set of local curvature estimates, we can locally modify the Lennard-Jones potential to concentrate more particles in areas of high curvature. Currently, we do this by associating a *scale* with each particle which is used in the computation of r in (7), i.e., $r = s\|\mathbf{r}_{ij}\|$ (this scale also affects other potentials that are written in terms of local coordinates). We define a force on each particle's scale estimate which is proportional to the magnitude of the curvature, $\sqrt{\kappa_1^2 + \kappa_2^2}$, and also add a restoring force which tries to equilibrate scale estimates between neighbors. The behavior of our curvature-dependant re-sampling is shown in Figure 8 and is qualitatively similar to the results presented in [26].

4.5. Oriented Curve Particles

To model curves with with oriented particles, the coordinate frame for each particle is the Frenet frame, i.e., each particle $\mathbf{p}_i$ has an associated tangent $\mathbf{t}_i$, main normal $\mathbf{m}_i$, and binormal $\mathbf{b}_i$. To locally control the curvature and torsion of space curves, we create a potential proportional to the squared error in the Frenet approximation (3),

$$\begin{aligned} \phi_{\rm L}(\mathbf{t}_i, \mathbf{m}_i, \mathbf{b}_i, \mathbf{r}_{ij}) &= \|\mathbf{r}_{ij} - x\mathbf{t}_i - \tfrac{1}{2}\kappa x^2 \mathbf{m}_i - \tfrac{1}{6}\kappa\tau x^3 \mathbf{b}_i\|^2 \psi(\|\mathbf{r}_{ij}\|) \\ &= [(y - \tfrac{1}{2}\kappa x^2)^2 + (z - \tfrac{1}{6}\kappa\tau x^3)^2]\psi(\|\mathbf{r}_{ij}\|), \end{aligned} \tag{12}$$

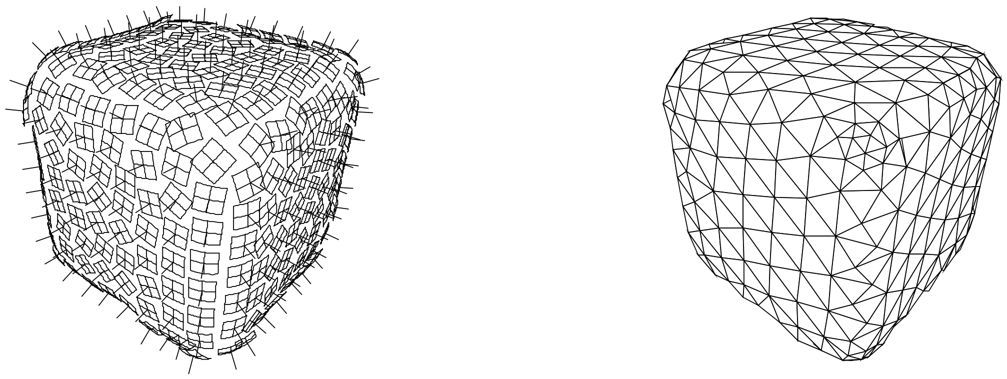


Figure 8: (a) Particles fit to surface of cube (principal frames are shown); (b) particles after curvature-dependent resampling (triangulation is shown).

where $x = \mathbf{t}_i \cdot \mathbf{r}_{ij}$, $y = \mathbf{m}_i \cdot \mathbf{r}_{ij}$, and $z = \mathbf{b}_i \cdot \mathbf{r}_{ij}$. We define a second potential based on the Frenet-Serret equations (2),

$$\phi_{\mathrm{M}}(\mathbf{t}_i, \mathbf{m}_i, \mathbf{b}_i, \mathbf{t}_j, \mathbf{m}_j, \mathbf{b}_j, \mathbf{r}_{ij}) = \\ (\|\mathbf{t}_i + \kappa x \mathbf{m}_i - \mathbf{t}_j\|^2 + \|\mathbf{m}_i - \kappa x \mathbf{t}_i + \tau x \mathbf{b}_i - \mathbf{m}_j\|^2 + \|\mathbf{b}_i - \tau x \mathbf{m}_i - \mathbf{b}_j\|^2)\psi(\|\mathbf{r}_{ij}\|). \tag{13}$$

5. Particle Dynamics

Having defined the internal energy associated with our system, we can derive its equations of motion. The derivative of the inter-particle potential with respect to the particle position and orientations gives rise to forces acting on the positions and torques acting on the orientations (which we represent using unit quaternions). The formulas for the inter-particle forces $\mathbf{f}_{ij}$ and torques $\boldsymbol{\tau}_{ij}$ are given in [18, 19]. The standard Newtonian equations of motion are then integrated using an explicit Euler's method [18, 19], which is equivalent to energy minimization with gradient descent.

A straightforward evaluation of the forces and torques at all of the particles requires $O(N^2)$ computation, where N is the number of particles. For large values of N, this can be prohibitively expensive. This computation has been shown to be reducible to $O(N \log N)$ time by hierarchical structuring of the particles [2]. In our work, we use a *k-d tree* [14] to subdivide space so that we can efficiently find all the particle's neighbors within some radius (usually $3\, r_0$, where r_0 is the natural inter-particle spacing). To further reduce computation, we perform this operation only occasionally and cache the list of neighbors for intermediate time steps.

6. Triangulation and Interpolation

Because our particle system does not give us an explicitly triangulated surface, we have developed an algorithm for triangulating particles. A commonly used technique for triangulating a 2-D surface or a 3-D volume is the Delaunay triangulation [4, 13]. In 2-D, a triangle is part of the Delaunay triangulation if no other vertices are within the circle circumscribing the triangle. To extend this idea to 3-D, we check the smallest sphere circumscribing each triangle (a 3-D analogue of the Gabriel graph [13]). We also limit the length of valid triangle edges (to 2 units, by default). This heuristic works well in practice when the surface is adequately sampled with respect to the curvature and the triangles are roughly equilateral. For curve particles, we simply compute the Gabriel graph, i.e., we check the smallest sphere circumscribing each pair of particles to see if it contains any other particles.

To better visualize the resulting surface, Gouraud, Phong, or flat shading can be applied to each triangle. For a smoother surface, a cubic patch can be interpolated at each triangle, since we know the normals at each corner. The major benefit of smoothly interpolating the surface across each triangle is that we can compute local differential geometric quantities [9] and support a finite element analysis on the patch deformation energies [22, 5, 11]. Many techniques have been developed to perform this interpolation, with different degrees of continuity and complexity [10]. In our current surface modeling system, we use cubic Bézier curves and triangles with a novel set of simple rules for placing the intermediate control points, as described below.

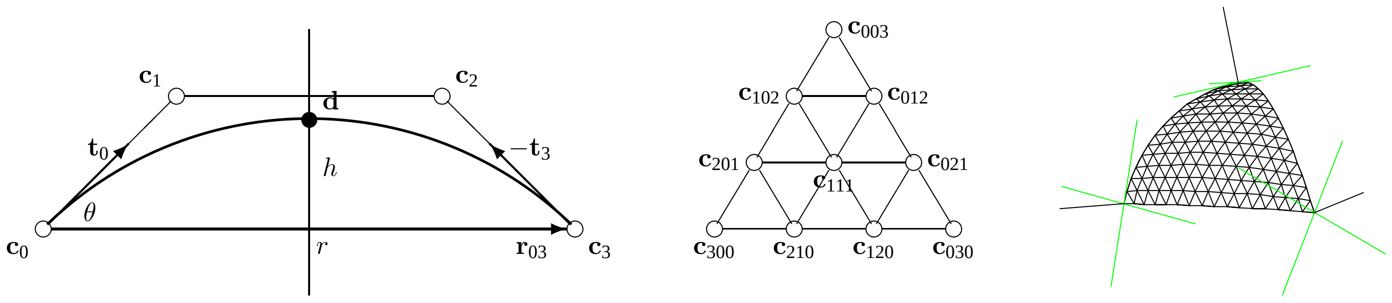


Figure 9: (a) Bézier boundary curve computation based on a symmetric circular arc fit; (b) B ézier triangle vertex numbering; (c) Cubic Bézier triangle fitted to a triplet of normal vectors (planar edge curves)

6.1. Cubic Bézier curves

A Bézier curve is polynomial parametric curve $\mathbf{x}(t)$ whose shape is controlled by the location of some *B ézier control points* $\{\mathbf{c}_i\}$.[2] The curve is computed as a *blend* of the control points,

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(t), \quad t \in [0,1], \qquad \text{where} \qquad B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \tag{14}$$

are *Bernstein polynomials*, and n is the *order* of the B ézier curve [7]. The control vertices $\mathbf{c}_0$ and $\mathbf{c}_n$ determine the endpoints of the curve, while the vectors $\mathbf{c}_1 - \mathbf{c}_0$ and $\mathbf{c}_n - \mathbf{c}_{n-1}$ determine the tangent directions at the endpoints (Figure 9a). In this paper, we use cubic Bézier curves ($n = 3$), since this is the minimum order required to allow explicit tangent direction control.

We now present a simple technique for placing the intermediate control points of a cubic B ézier curve, $\{\mathbf{c}_1, \mathbf{c}_2\}$, given two endpoints and tangents $\{\mathbf{c}_0, \mathbf{c}_3, \mathbf{t}_0, \mathbf{t}_3\}$ (Figure 9). To match the tangent directions, we must have $\mathbf{c}_1 = \mathbf{c}_0 + s_0\mathbf{t}_0$ and $\mathbf{c}_2 = \mathbf{c}_3 - s_3\mathbf{t}_3$. To compute s_0, we assume that the two tangent directions $\mathbf{t}_0$ and $\mathbf{t}_3$ are anti-symmetric ($\mathbf{t}_0 \times \mathbf{r}_{01} = -\mathbf{t}_3 \times \mathbf{r}_{01}$) and that the curve is actually part of a circle. We choose the value of s_0 such that the center position along the B ézier curve $\mathbf{d}$ lies on the circular arc through the two endpoints $\mathbf{c}_0$ and $\mathbf{c}_3$ and tangent to the two edge directions $\mathbf{t}_0$ and $\mathbf{t}_3$ (Figure 9a). The height at the midpoint of the circular arc is $h = r \sin\theta / (2 + 2\cos\theta)$, where $r = |\mathbf{r}_{01}|$ and $\cos\theta = \mathbf{t}_0 \cdot \mathbf{r}_{01}/r$, whereas the height at the midpoint of the Bézier curve is $h = {}^3\!/_4(s_0 \sin\theta)$, from which we obtain $s_0 = \frac{2r}{3(1+\cos\theta)}$. This equation is applied independently to each endpoint ($\mathbf{c}_0$ and $\mathbf{c}_3$). Thus, while for the analysis we assume that the curve is part of a symmetric circular arc, in practice, each tangent can vary independently.

Figure 10a shows an example of our cubic B ézier interpolant applied to a collection of Frenet frames. This curve behaves like a 3-D "snake" with local tangent direction control. To make the snake active or deformable, internal energies must be added. This can be accomplished either by directly computing finite-element energies over the cubic curve segments (e.g., by numerical quadrature), or by using the oriented particle potentials.

6.2. An Approximately G^1 Bézier Triangular Patch

A number of techniques have been developed to smoothly interpolate a triangulated collection of 3-D points with associated normals and curvatures [10, 6, 12]. The approach used in this paper uses only the normal information at each point and is based on the cubic Bézier triangle (as in [6]). The B ézier triangle can be written in *barycentric coordinates* (u, v, w) [7] as

$$\mathbf{x}(u,v,w) = \sum_{i+j+k=n} \mathbf{c}_{ijk} B_{ijk}^n(u,v,w), \qquad \text{with} \qquad B_{ijk}^n(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k. \tag{15}$$

An example of the control points for a cubic B ézier triangle is shown in Figure 9b.

To satisfy the normality condition at each corner of the triangle, the two adjacent B ézier control points must lie in the tangent plane perpendicular to the normal. To compute the location of these edge points, we use the technique we developed previously for Bézier curves after first computing tangent directions for each edge. This is achieved by computing the intersection of the surface tangent plane with the plane containing $\mathbf{r}$, the line joining the two vertices, and either (1) the vertex normal or (2) the average normal between the two vertices on a given edge. The latter choice, which results in planar edge curves, is the

[2]The more standard notation is $\mathbf{b}_i$, but we are already using $\mathbf{b}$ for the binormal.

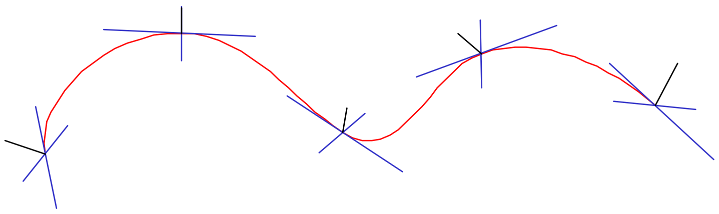
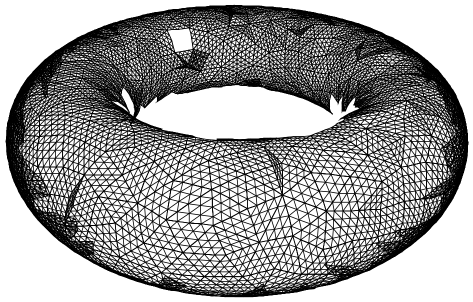



Figure 10: (a) Cubic Bézier interpolant applied to a collection of Frenet frames. (b) B ézier triangles fitted to triangulated torus data (only forward facing triangels are rendered)

choice made in [6]. The tangent vector is thus $\mathbf{t}_0 = \pm\mathcal{N}(\mathbf{n}_0 \times \mathbf{n}_e)$, and $\mathbf{n}_e = (\mathbf{n}_0 + \mathbf{n}_1) \times \mathbf{r}$ for the planar edge curve case or $\mathbf{n}_e = \mathbf{n}_0 \times \mathbf{r}$ for the non-planar case, with the sign of $\mathbf{t}_0$ chosen such that $\mathbf{t}_0 \cdot \mathbf{r} > 0$.

To compute the central Bézier control point (Figure 9b), we extrapolate each triplet of control vertices at each corner, e.g., $\mathbf{c}_{111}^{(0)} = \mathbf{c}_{300} + \alpha(\mathbf{c}_{210} - \mathbf{c}_{300}) + \alpha(\mathbf{c}_{201} - \mathbf{c}_{300})$ with $\alpha = 3/4$ and take the average of all three extrapolated center points $\mathbf{c}_{111} = (\mathbf{c}_{111}^{(0)} + \mathbf{c}_{111}^{(1)} + \mathbf{c}_{111}^{(2)})/3$. This is similar to taking the intersection of the three normal planes as suggested by [6][3], but is simpler and more stable. The result of applying our technique to a triplet of oriented frames is shown in Figure 9c. The result of using our technique to interpolate and render a collection of points sampled on a torus is shown in Figure 10b.

7. Discussion

We have developed a particle-based model of deformable surfaces and applied it to several computer vision problems. Our new model, which is based on oriented particles with new interaction potentials, has characteristics of both physically-based surface models and of particle systems. It can be used to model smooth, elastic, moldable surfaces, like traditional splines, and it allows for arbitrary interactions and topologies, like particle systems.

The particle-based surface model we have developed has a number of advantages over traditional spline-based surface models. Particle-based surfaces are easy to shape, extend, join, and separate. By adjusting the relative strengths of various potential functions, the surface's resistance to stretching, bending, or variation in curvature can all be controlled. The topology of particle-based surfaces can easily be modified, as can the sampling density, and surfaces can be fitted to arbitrary collections of 3-D data points.

Our particle-based surface model shares some characteristics with local patch models such as those developed by Sander and Zucker [15]. Our particles interact in a manner similar to their patches, but in addition, particle positions are not fixed, so that a more even distribution of samples can be achieved. More importantly, the triangulation process ensures that a globally consistent smooth analytic surface is defined at all times. This enables us to derive interaction potentials based on finite element analysis, so that arbitrary smoothness conditions or material properties can be simulated.

A limitation of particle-based surfaces is that it is harder to achieve exact analytic (mathematical) control over the shape of the surface. For example, the reconstructed torus is not circularly symmetric, due to the discretization effects of the relatively small number of particles. This behavior could be remedied by adding constraints in the form of extra potentials, e.g., a circular symmetry potential for the torus. Particle-based surfaces also require more computation to simulate their dynamics than spline-based surfaces; the latter may therefore be more appropriate when shape flexibility is not paramount. One could easily envision a hybrid system where spline or other parametric surfaces co-exist with particle-based surfaces, using each system's relative advantages where appropriate.

In future work, we plan merge particle-based curves and surfaces into a single system where curves can lie on surfaces to model creases, terminations, and surface markings. Our current system allows us to model creases using curve particle chains, but does not represent the multiple surface normals that exist at such creases. Curves representing surface markings can be implemented either by making certain curve and surface particles coincident, or by allowing curves to "float" on top of surfaces while maintaining consistency constraints. If successful, this representation has the potential to be an extremely powerful tool for computer vision since both shape and photometric effects (albedo variation) could be represented simultaneously.

We also plan to explore in more detail the additional flexibility available with local curvature estimates. For example, once a surface has adapted to an initial CAD model, local curvature estimates could be used to implement a kind of "shape memory"

[3]Actually, [6] use a formula involving *offset* normal planes.

even in the absence of the original external forces. Careful attention would have to be paid to our implementation of the discrete differential geometry since, unlike in the curve case, local curvature estimates are not sufficient to completely characterize the shape of a surface [9].

Our approach is very promising in the long run, since we can model highly detailed shape information while maintaining the desired level of continuity and smoothness, and we can also model (and reconstruct) surfaces of arbitrary topology. Combined with a better understanding of the local differential geometry of surfaces and the concurrent modeling of curves and surfaces, we believe these techniques will form the basis of a powerful new approach to shape reconstruction in computer vision and other domains.

References

[1] G. J. Agin and T. O. Binford. Computer description of curved objects. *IEEE Transactions on Computers*, C-25(4):439–449, April 1976.

[2] A. Appel. An efficient algorithm for many-body simulations. *SIAM J. Sci. Stat. Comput.*, 6(1), 1985.

[3] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, Cambridge, Massachusetts, 1987.

[4] J.-D. Boissonat. Representing 2D and 3D shapes with the Delaunay triangulation. In *Seventh International Conference on Pattern Recognition (ICPR'84)*, pages 745–748, Montreal, Canada, July 1984.

[5] G. Celniker and D. Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics (SIGGRAPH'91)*, 25(4):257–266, July 1991.

[6] T. DeRose and S. Mann. An approximately G^1 cubic surface interpolant. In *Mathematical Methods in Computer Aided Design II*, pages 185–196. Academic Press, San Diego, 1992.

[7] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, Boston, Massachusetts, 3rd edition, 1992.

[8] R. W. Hockney and J. W. Eastwood. *Computer Simulation using Particles*. McGraw-Hill Inc., New York, 1988.

[9] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, Massachusetts, 1990.

[10] M. Lounsbery, S. Mann, and T. DeRose. Parametric surface interpolation. *IEEE Computer Graphics and Applications*, 12(5):45–52, September 1992.

[11] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Fourth International Conference on Computer Vision (ICCV'93)*, Berlin, Germany, May 1993. IEEE Computer Society Press.

[12] H. P. Moreton and C. H. Séquin. Functional optimization for fair surface design. *Computer Graphics (SIGGRAPH'92)*, 26(2):167–176, July 1992.

[13] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Academic Press, New York, 1985.

[14] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, Massachusetts, 1989.

[15] P. T. Sander and S. W. Zucker. Inferring surface trace and differential structure from 3-D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):833–854, September 1990.

[16] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, February 1990.

[17] R. Szeliski. Bayesian modeling of uncertainty in low-level vision. *International Journal of Computer Vision*, 5(3):271–301, December 1990.

[18] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. Technical Report 91/14, Digital Equipment Corporation, Cambridge Research Lab, December 1991.

[19] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. *Computer Graphics (SIGGRAPH'92)*, 26(2):185–194, July 1992.

[20] R. Szeliski, D. Tonnesen, and D. Terzopoulos. Modeling surfaces of arbitrary topology with dynamic particles. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, New York, New York, June 1993.

[21] D. Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-10(4):417–438, July 1988.

[22] D. Terzopoulos and D. Metaxas. Dynamic 3D models with local and global deformations: Deformable superquadrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7):703–714, July 1991.

[23] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Rcovering 3D shape and nonrigid motion. *Artificial Intelligence*, 36(1):91–123, August 1988.

[24] D. Terzopoulos, J. Platt, and K. Fleischer. Heating and melting deformable models (From Goop to Glop). In *Proceedings Graphics Interface*, pages 219–226. Graphics Interface, June 1989.

[25] D. Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface '91*, pages 255–262, 1991.

[26] G. Turk. Re-tiling polygonal surfaces. *Computer Graphics (SIGGRAPH'92)*, 26(2):55–64, July 1992.

[27] B. C. Vemuri, A. Mitiche, and J. K. Aggarwal. Curvature-based representation of objects from range data. *Image and Vision Computing*, 4:107–114, 1986.