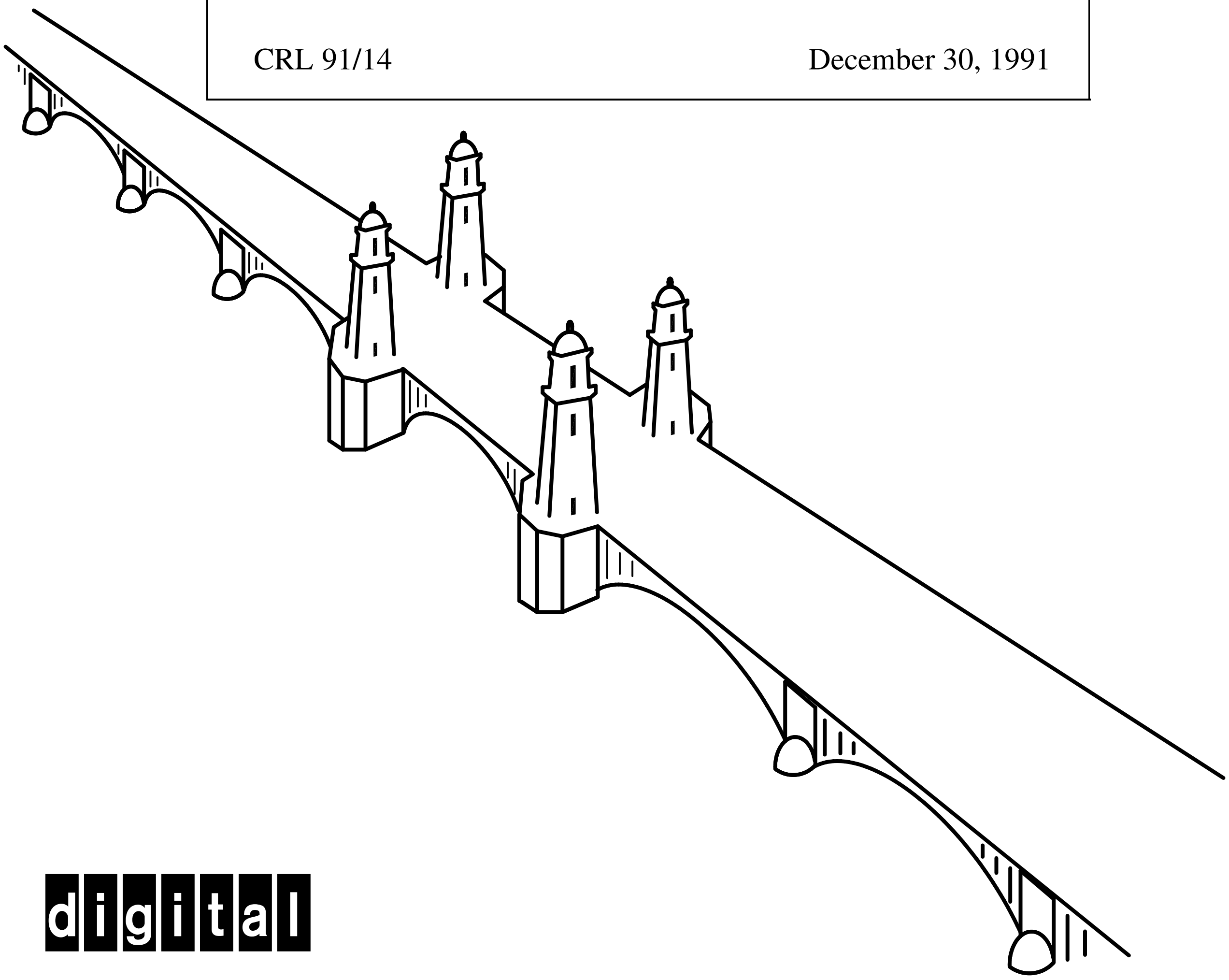
Surface Modeling with Oriented Particle Systems

Richard Szeliski and David Tonnesen

Digital Equipment Corporation

Cambridge Research Lab

CRL 91/14 December 30, 1991



digital

CAMBRIDGE RESEARCH LABORATORY

Technical Report Series

Cambridge Research Laboratory
One Kendall Square
Cambridge, Massachusetts 02139

Surface Modeling with Oriented Particle Systems

Richard Szeliski and David Tonnesen[1]

Digital Equipment Corporation

Cambridge Research Lab

CRL 91/14 December 30, 1991

Abstract

Splines and deformable surface models are widely used in computer graphics to describe free-form surfaces. These methods require manual preprocessing to discretize the surface into patches and to specify their connectivity. We present a new model of elastic surfaces based on interacting particle systems, which, unlike previous techniques, can be used to split, join, or extend surfaces without the need for manual intervention. The particles we use have long-range attraction forces and short-range repulsion forces and follow Newtonian dynamics, much like recent computational models of fluids and solids. To enable our particles to model surface elements instead of point masses or volume elements, we add an orientation to each particle's state. We devise new interaction potentials for our *oriented particles* which favor locally planar or spherical arrangements. We also develop techniques for adding new particles automatically, which enables our surfaces to stretch and grow. We demonstrate the application of our new particle system to modeling surfaces in 3-D and the interpolation of 3-D point sets.

Keywords: Computer graphics, geometric modeling, surface interpolation, particle systems, physically-based modeling, oriented particles, dynamics, simulation.

[1]Dept. of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

Contents

1 Introduction

The modeling of free-form surfaces is one of the central issues of computer graphics. Spline models [Bartels *et al.*, 1987; Farin, 1990] and deformable surface models [Terzopoulos *et al.*, 1987b; Terzopoulos and Fleischer, 1988a] have been very successful in creating and animating such surfaces. However, these methods either require the discretization of the surface into patches (for spline surfaces) or the specification of local connectivity (for spring-mass systems). These steps can involve a significant amount of manual preprocessing before the surface model can be used.

For shape design and rapid prototyping applications, we require a highly interactive system which does not force the designer to think about the underlying representation or be limited by its choice [Sachs *et al.*, 1991]. For example, we require the basic abilities to join several surfaces together, to split surfaces along arbitrary lines, or to extend existing surfaces, without specifying exact connectivity. For scientific visualization, data interpretation, and robotics applications, we require a modeling system that can interpolate a set of scattered 3-D data without knowing the topology of the surface. To construct such a system, we will keep the ideas of deformation energies from elastic surface models, but use interacting particles to build our surfaces.

Particle systems have been used in computer graphics by Reeves [1983] and Sims [1990] to model natural phenomena such as fire and waterfalls. In these models, particles move under the influence of force fields and constraints but do not interact with each other. More recent particle systems borrow ideas from molecular dynamics to model liquids and solids [Miller and Pearce, 1989; Terzopoulos *et al.*, 1989; Tonnesen, 1991]. In these models, which have spherically symmetric potential fields, particles arrange themselves into volumes rather than surfaces.

In this paper, we develop a new kind of particle model, *oriented particles*, which overcome this natural tendency to form solids and prefer to form surfaces instead. Each particle has a local coordinate frame which is updated during the simulation. We design new interaction potentials which favor locally planar or locally spherical arrangements of particles. These interaction potentials are used in conjunction with more traditional long-range attraction forces and short-range repulsion forces which control the average inter-particle spacing.

Our new surface model thus shares characteristics of both deformable surface models and particle systems. Like traditional spline models, it can be used to model free-form surfaces and to smoothly interpolate sparse data. Like interacting particle models of solids and liquids, our surfaces can be split, joined, or extended without the need for re-parameterization or manual intervention. We can thus use our new technique as a tool for modeling a wider range of surface shapes.

The remainder of the paper is organized as follows. In Section 2 we review traditional splines and deformable surface models. We also review particle systems and present the potential functions traditionally used in molecular dynamics. In Section 3 we present our new oriented particle model and the new interaction potentials which favor locally planar and locally spherical arrangements. Section 4 presents the dynamics (equations of motion) associated with our interacting particle system and discusses numerical time integration and complexity issues. Section 5 discusses alternative rendering techniques for particles and surfaces. In Section 6 we present simple shaping operations for surfaces built out of particles. In Section 7 we show how to extend existing surfaces by adding new particles, and how to use this approach to automatically fit surfaces to 3-D point collections. In Section 8 we discuss applications of our system to geometric modeling. We conclude with a discussion of the merits and drawbacks of oriented particle systems and a summary of our contributions.

2 Background

Our new surface modeling technique is based on two previously separate areas of computer graphics, namely deformable surface models and particle systems. Below, we present a brief review of these two fields.

2.1 Deformable Surface Models

Traditional spline techniques [Bartels *et al.*, 1987; Farin, 1990] model an object's surface as a collection of piecewise-polynomial patches, with appropriate continuity constraints imposed between the patches to achieve the desired degree of smoothness. Within a particular patch, the surface's shape can be expressed using a superposition of basis functions

$$\mathbf{s}(u_1, u_2) = \sum_i \mathbf{v}_i B_i(u_1, u_2) \tag{1}$$

where $\mathbf{s}(u_1, u_2)$ are the 3D coordinates of the surface as a function of the underlying parameters (u_1, u_2), $\mathbf{v}_i$ are the *control vertices*, and $B_i(u_1, u_2)$ are the piecewise polynomial *basis functions*. The surface shape can then be adjusted by interactively positioning the control vertices or by directly manipulating points on the surface [Bartels and Beatty, 1989].

Elastically deformable surface models [Terzopoulos *et al.*, 1987b; Terzopoulos and Fleischer, 1988a] also start with a parametric representation for the surface $\mathbf{s}(u_1, u_2)$. To define the dynamics of the surface, Terzopoulos *et al.* [1987b] use the metric tensor or first fundamental form $\mathbf{G}$ whose components are given by

$$G_{ij}(\mathbf{s}(u_1, u_2)) = \frac{\partial \mathbf{s}}{\partial u_i} \cdot \frac{\partial \mathbf{s}}{\partial u_j} \tag{2}$$

and the curvature tensor or second fundamental form $\mathbf{B}$ whose components are given by

$$B_{ij}(\mathbf{s}(u_1, u_2)) = \mathbf{n} \cdot \frac{\partial^2 \mathbf{s}}{\partial u_i \partial u_j}, \tag{3}$$

where $\mathbf{n}$ is the unit surface normal. A simplified deformation energy

$$\mathcal{E}(\mathbf{s}) = \int_{\Omega} \sum_{i,j=1}^{2} (\eta_{ij}(G_{ij} - G_{ij}^0)^2 + \xi_{ij}(B_{ij} - B_{ij}^0)^2)\, du_1\, du_2, \qquad (4)$$

where $\eta_{ij}(u_1, u_2)$ and $\xi_{ij}(u_1, u_2)$ are weighting functions and G_{ij}^0 and B_{ij}^0 are the rest lengths and rest curvatures, is used to compute the elastic restoring forces for the surface [Terzopoulos *et al.*, 1987b]. Additional forces to model gravity, external spring constraints, viscous drag, and collisions with impenetrable objects can then be added.

To simulate the deformable surface, these analytic equations are discretized using either finite element or finite difference methods. This results in a set of coupled differential equations governing the temporal evolution of the set of control points. Physically-based surface models can be thought of as adding temporal dynamics and elastic forces to an otherwise inert spline model. They can also be thought of as a collection of point masses connected with a set of finite-length springs [Terzopoulos *et al.*, 1989].

Physically-based surface models have been used to model a wide variety of materials, including cloth [Weil, 1986; Breen *et al.*, 1991], membranes [Terzopoulos *et al.*, 1987b], and paper [Terzopoulos and Fleischer, 1988b]. Viscoelasticity, plasticity, and fracture have been incorporated to widen the range of modeled phenomena [Terzopoulos and Fleischer, 1988b].

The main drawback of both splines and deformable surface models is that the rough shape of the object must be known or specified in advance [Terzopoulos *et al.*, 1987a]. For spline models, this means discretizing the surface into a collection of patches with appropriate continuity conditions, which is generally a difficult problem [Loop and DeRose, 1990]. For deformable surface models, we can bypass the patch formation stage by specifying the location and interconnectivity of the point masses in the finite element approximation. In either approach, defining the model topology in advance remains a tedious process. Furthermore, it severely limits the flexibility of a given surface model. For example, if the surface is originally a sphere, it is impossible to deform it into an object

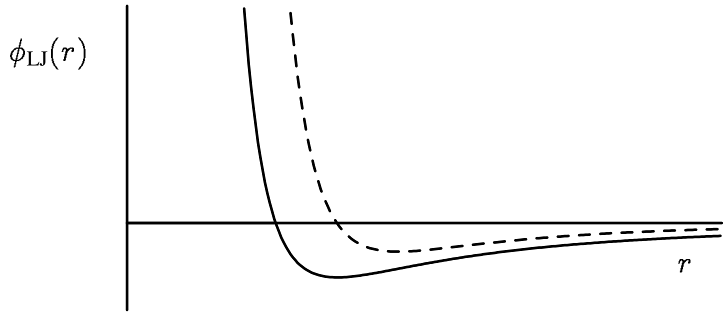


Figure 1: Lennard-Jones type function, $\phi_{\mathrm{LJ}}(r) = B/r^n - A/r^m$. The solid line shows the potential function $\phi_{\mathrm{LJ}}(r)$, and the dashed line shows the force function $f(r) = -\frac{d}{dr}\phi_{\mathrm{LJ}}(r)$.

of a different topology such as a torus.

2.2 Particle Systems

Particle systems consist of a large number of point masses (particles) moving under the influence of external forces such as gravity, vortex fields, and collisions with stationary obstacles. Each particle is represented by its position, velocity, acceleration, mass, and other attributes such as color. The ensemble of particles moves according to Newton's laws of motion. Particle systems built from non-interacting particles have been used to realistically model a range of natural phenomena including fire [Reeves, 1983] and waterfalls [Sims, 1990].

Ideas from molecular dynamics have been used to develop models of interacting particles [Terzopoulos *et al.*, 1989; Miller and Pearce, 1989; Tonnesen, 1991]. 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 ϕ_{LJ} shown in Figure 1. The force $\mathbf{f}_{ij}$ attracting a molecule to its neighbor is computed from the derivative of the potential function

$$\mathbf{f}_{ij} = -\nabla_{\mathbf{r}}\phi_{\mathrm{LJ}}(\|\mathbf{r}_{ij}\|), \tag{5}$$

where $\mathbf{r}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ is the vector distance between molecules i and j (Figure 2).

Physical systems whose dynamics are governed by potential functions and damping will evolve towards lower energy states. When external forces are insignificant, molecules will arrange themselves into closely packed structures to minimize their total energy. For circularly symmetric potential energy functions in 2-D, the molecules will arrange themselves into hexagonal orderings. In 3-D, the molecules will arrange themselves into hexagonally ordered 2-D layers, and therefore make good models of deformable solids [Tonnesen, 1991]. When external forces become larger or internal particle forces smaller, the behavior resembles that of viscous fluids [Terzopoulos *et al.*, 1989; Miller and Pearce, 1989]. More sophisticated models of molecular dynamics are used in simulations of physics and chemistry [Hockney and Eastwood, 1988]; however, these are designed for high accuracy and are usually too slow for animation or modeling applications.

3 Oriented Particles

While particle systems are much more flexible than deformable surface models in arranging themselves into arbitrary shapes and topologies, they do suffer from one major drawback: in the absence of external forces and constraints, 3-D particle systems prefer to arrange themselves into solids rather than surfaces. To overcome this limitation, we introduce a new distributed model of surface shape which we call *oriented particles*, in which each particle represents a small surface element (which we could call a “surfel”). In addition to having a position, an oriented particle also has its own local coordinate frame, which adds three new degrees of freedom to each particle’s state.

To force oriented particles to group themselves into surface-like arrangements, we devise a collection of new potential functions. These potential functions can be derived from the deformation energies of local triangular patches using finite element analysis. We defer the details of this analysis to Appendix A, and present below a more intuitive explanation

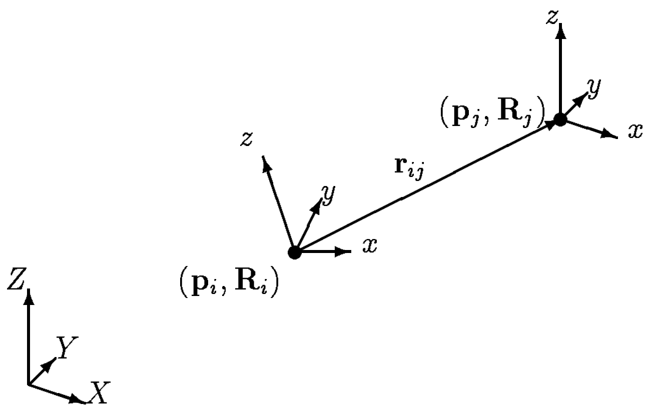


Figure 2: Global and local coordinate frames. The global interparticle distance $\mathbf{r}_{ij}$ is computed from the global coordinates $\mathbf{p}_i$ and $\mathbf{p}_j$ of particles i and j. The local distance $\mathbf{d}_{ij}$ is computed from $\mathbf{r}_{ij}$ and the rotation matrix $\mathbf{R}_i$.

based on analogies with physical surfaces and oriented particles such as magnetic dipoles.

Each oriented particle defines both a normal vector (z in Figure 2) 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 and $\mathbf{R}_i$ is a 3×3 rotation matrix which defines the orientation of its local coordinate frame (relative to the global frame (X, Y, Z)). The third column of $\mathbf{R}_i$ is the local normal vector $\mathbf{n}_i$.

For surfaces whose rest (minimum energy) configurations are flat planes, we would expect neighboring particles to lie in each other's tangent planes. We can express this *co-planarity* condition as

$$\phi_{\mathrm{P}}(\mathbf{n}_i, \mathbf{r}_{ij}) = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \psi(\|\mathbf{r}_{ij}\|), \tag{6}$$

i.e., the energy is proportional to the dot product between the surface normal and the vector to the neighboring particle (Figure 3a). The weighting function $\psi(r)$ is a monotone decreasing function used to limit the range of inter-particle interactions.

The co-planarity condition does not control the "twist" in the surface between two

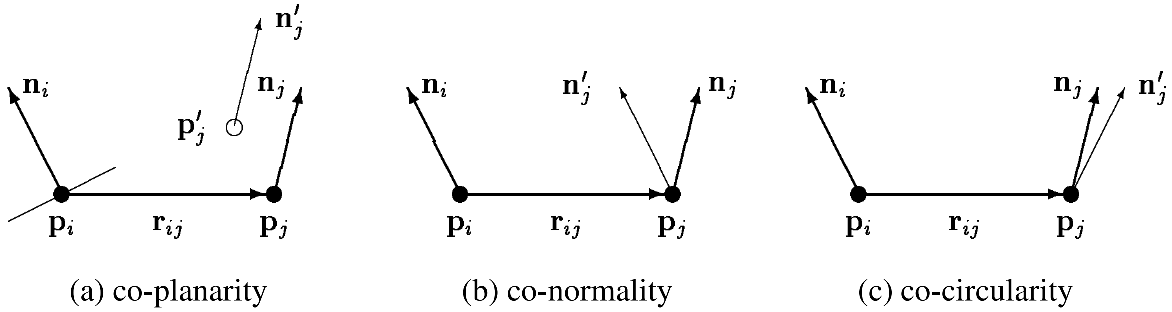


Figure 3: The three oriented particle interaction potentials. The open circles and thin arrows indicate a possible new position or orientation for the second particle which would lead to a null potential.

particles. To limit this, we introduce a *co-normality* potential

$$\phi_{\mathrm{N}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \psi(\|\mathbf{r}_{ij}\|), \tag{7}$$

which attempts to line up neighboring normals, much like interacting magnetic dipoles (Figure 3b).

An alternative to surfaces which prefer zero curvature or local planarity are surfaces which favor constant curvatures. This can be enforced with a *co-circularity* potential

$$\phi_{\mathrm{C}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) = ((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij})^2 \psi(\|\mathbf{r}_{ij}\|) \tag{8}$$

which is zero when normals are antisymmetrical with respect to the vector joining two particles (Figure 3c). This is the natural configuration for surface normals on a sphere.

The above potentials can also be written in term of a particle's *local coordinates*, e.g., by replacing the interparticle distance $\mathbf{r}_{ij}$ by

$$\mathbf{d}_{ij} = \mathbf{R}_i^{-1} \mathbf{r}_{ij} = \mathbf{R}_i^{-1} (\mathbf{p}_j - \mathbf{p}_i), \tag{9}$$

which gives the coordinates of particle j in particle i's local coordinate frame. This not only simplifies certain potential equations such as (6), but also enables us to write use a weighting function $\psi(\mathbf{d}_{ij})$ which is not circularly symmetric, e.g., one which weights particles more if they are near a given particle's tangent plane. In practice, we use

$$\psi(x, y, z) = K \exp\left(-\frac{x^2 + y^2}{2a^2} - \frac{z^2}{2b^2}\right) \tag{10}$$

with $b \leq a$.

To control the bending and stiffness characteristics of our deformable surface, we use a weighted sum of potential energies

$$E_{ij} = \alpha_{\mathrm{LJ}}\phi_{\mathrm{LJ}}(\|\mathbf{r}_{ij}\|) + \alpha_{\mathrm{P}}\phi_{\mathrm{P}}(\mathbf{n}_i, \mathbf{r}_{ij}) + \alpha_{\mathrm{N}}\phi_{\mathrm{N}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) + \alpha_{\mathrm{C}}\phi_{\mathrm{C}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}). \tag{11}$$

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 surfaces tendency towards uniform local curvature. The total internal energy of the system E_{int} is computed by summing the inter-particle energies

$$E_{\mathrm{int}} = \sum_i \sum_j E_{ij}.$$

4 Particle Dynamics

Having defined the internal energy associated with our system, we can derive its equations of motion. The variation of 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. The formulas for the inter-particle forces $\mathbf{f}_{ij}$ and torques $\boldsymbol{\tau}_{ij}$ are given in Appendix B. These forces and torques can be summed over all interacting particles to obtain

$$\mathbf{f}_i = \sum_{j \in \mathcal{N}_i} \mathbf{f}_{ij} + \mathbf{f}_{\mathrm{ext}}(\mathbf{p}_i) - \beta_0 \mathbf{v}_i, \tag{12}$$

$$\boldsymbol{\tau}_i = \sum_{j \in \mathcal{N}_i} \boldsymbol{\tau}_{ij} - \beta_1 \boldsymbol{\omega}_i, \tag{13}$$

where $\mathcal{N}_i$ are the neighbors of i (Section 4.2). Here, we have lumped all external forces such as gravity, user-defined control forces, and non-linear constraints into $\mathbf{f}_{\mathrm{ext}}$, and added velocity-dependent damping $\beta_0 \mathbf{v}_i$ and $\beta_1 \boldsymbol{\omega}_i$.

Using these forces and torques, we can write the standard Newtonian equations of motion

$$\begin{aligned} \mathbf{a}_i &= \mathbf{f}_i/m_i & \boldsymbol{\alpha}_i &= \mathbf{I}_i^{-1}\boldsymbol{\tau}_i \\ \dot{\mathbf{v}}_i &= \mathbf{a}_i & \dot{\boldsymbol{\omega}}_i &= \boldsymbol{\alpha}_i \\ \dot{\mathbf{p}}_i &= \mathbf{v}_i & \dot{\mathbf{q}}_i &= \boldsymbol{\omega}_i, \end{aligned}$$

where m_i is the particle's mass, and $\mathbf{I}_i$ is its rotational inertia (which for a circularly symmetric particle is diagonal). The equations for translational acceleration $\mathbf{a}$, velocity $\mathbf{v}$, and position $\mathbf{p}$ are the same as those commonly used in physically-based modeling and particle systems. The equations for rotational acceleration $\boldsymbol{\alpha}$, velocity $\boldsymbol{\omega}$, and orientation $\mathbf{q}$ are less commonly used. The rotational accelerations and velocities are vector quantities representing infinitesimal changes and can be added and scaled as regular vectors. The computation of the orientation (local coordinate frame) is more complex, and a variety of representations could be used. While we use the rotation matrix $\mathbf{R}$ to convert from local coordinates to global coordinates and vice versa, we use a unit quaternion $\mathbf{q}$ as the state to be updated. The unit quaternion

$$\mathbf{q} = (\mathbf{w}, s) \text{ with } \begin{aligned} \mathbf{w} &= \mathbf{n}\sin(\theta/2) \\ s &= \cos(\theta/2) \end{aligned}$$

represents a rotation of θ about the unit normal axis $\mathbf{n}$. To update this quaternion, we simply form a new unit quaternion from the current angular velocity $\boldsymbol{\omega}$ and the time step Δt, and use quaternion multiplication [Shoemake, 1985].

4.1 Numerical Time Integration

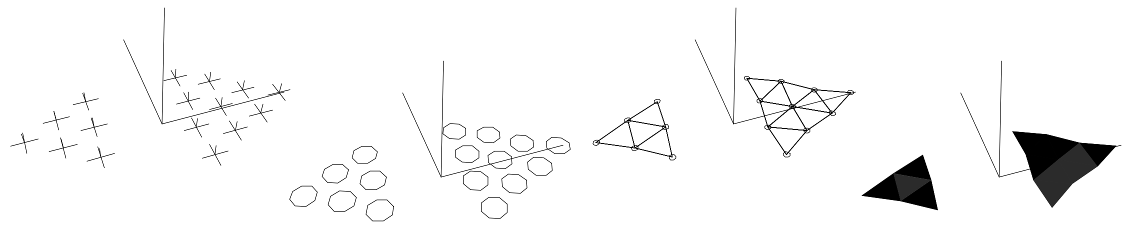
To simulate the dynamics of our particle system, we integrate the above system of differential equations through time. At each time step $t_{j+1} = t_j + \Delta t$ we sum all of the forces acting on each particle i and integrate over the time interval. The forces include the inter-particle forces, collision forces, gravity, and damping forces. We use Euler's method [Press *et al.*, 1988] to advance the current velocity and position over the time step. More sophisticated numerical integration techniques such as Runge-Kutta [Press *et al.*, 1988] or semi-implicit methods [Terzopoulos *et al.*, 1987b] could also be used.

4.2 Controlling Complexity

The straight forward evaluation of (12) and (13) to compute the forces and torques at all of the particles requires $O(N^2)$ computation. 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 data [Appel, 1985]. In our work, we use a *k-d tree* [Samet, 1989] to subdivide space sufficiently so that we can efficiently find all of a point's neighbors within some radius (e.g., 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.

5 Rendering

A variety of techniques have been developed for rendering particle systems, including light emitting points [Blinn, 1982a; Reeves and Blau, 1985; Sims, 1990] and iso-surfaces or "blobbies" [Blinn, 1982b; Wyvill *et al.*, 1986; Hersh, 1987; Miller and Pearce, 1989; Tonnesen, 1989] for modeling volumes. For rendering oriented particles, simple icons such as axes (Figure 4a) or flat discs (Figure 4b) can be used to indicate the location and orientation of each particle. A more realistic looking surface display requires the generation of a triangulation over our set of particles, which can then be displayed as a wireframe (Figure 4c) or shaded surface (Figure 4d). For shaded rendering, Gouraud, Phong, or flat shading can be applied to each triangle. For a smoother looking surface, a cubic patch can be created at each triangle (since we know the normals at each corner).



(a) (b) (c) (d)

Figure 4: Rendering techniques for particle-based surfaces: (a) axes, (b) discs, (c) wireframe triangulation (d) flat-shaded triangulation.

Because our particle system does not explicitly give us a triangulation of the surface, we have developed an algorithm for computing it. A commonly used technique for triangulating a surface in 2-D or a volume in 3-D is the Delaunay triangulation [Boissonat, 1984]. 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. This heuristic works well in practice when the surface is sufficiently sampled with respect to the curvature. The results of using our triangulation algorithm are shown in Figures 4c and 4d.

6 Basic modeling operations

This section describes some basic operations for interactively creating, editing, and shaping particle-based surfaces.[1]

The most basic operations are adding, moving, and deleting single particles. We can form a simple surface patch by creating a number of particles in a plane and allowing the

[1] While the figures in the printed version of this report are reproduced in black-and-white, the electronic version of this document (see inside front cover) contains color PostScript figures that can be viewed with a PostScript previewer.

system dynamics to adjust the particles into a smooth surface. We can enlarge the surface by adding more particles (either inside or at the edges), shape the surface by moving particles around or changing their orientation, or trim the surface by deleting particles. All particle editing uses direct manipulation. Currently, we use a 2-D locator (mouse) to perform 3-D locating and manipulation, inferring the missing depth coordinate when necessary from the depths of nearby particles. Adding 3-D input devices for direct 3-D manipulation [Sachs *et al.*, 1991] would be of obvious benefit.

To control the shape more accurately, we can fix the positions and/or orientations of individual particles. Figure 5 shows an example of two particle "chains" whose endpoints have been fixed in space.[2] When simulated gravity is turned on, the two chains fall together and join at the bottom due to inter-particle attraction forces.

In addition to particle-based surfaces, our modeling system also contains user-definable solid objects such as planes, spheres, cylinders, and arbitrary polyhedra. These objects are used to shape particle-based surfaces, by acting as solid tools [Platt and Barr, 1988; Witkin and Welch, 1990], as attracting surfaces, as "movers" which grab all of the particles inside them, or as large erasers. These geometric objects are positioned and oriented using the same direct manipulation techniques as are used with particles. Another possibility for direct particle or surface manipulation would be extended free-form deformations [Coquillart, 1990].

Using these tools, particle-based surfaces can be "cold welded" together by abutting their edges (Figure 6). Inter-particle forces pull the surfaces together and readjust the particle locations to obtain a seamless surface with uniform sampling density. We can "cut" a surface into two by separating it with a knife-like constraint surface (Figure 7). Here, we use the "heat" of the cutting tool to weaken the inter-particle bonds [Tonnesen, 1991]. Or we can "crease" a surface by designating a line of particles to be *unoriented*,

[2] We can form chains by restricting the particles to lie in the $y = 0$ plane. The particles then cluster into curved segments which behave much like the *snakes* of Kass, Witkin, and Terzopoulos [1988].

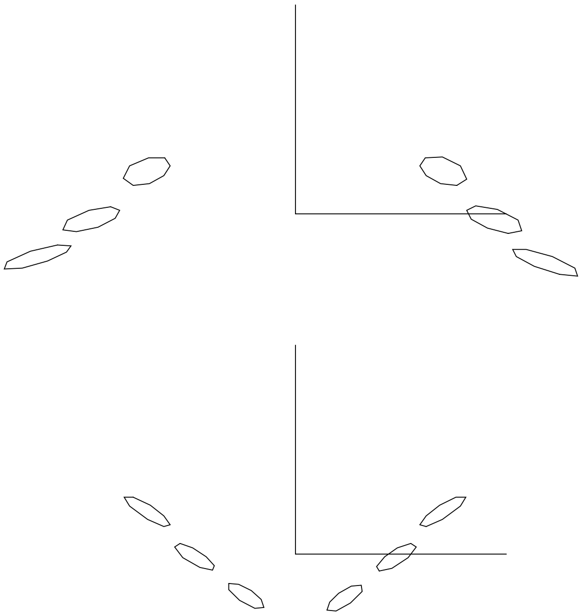


Figure 5: Chain linking in 2-D. The two chain pieces swing under gravity, and when their endpoints are near each other, they link into a single chain, like a trapeze.

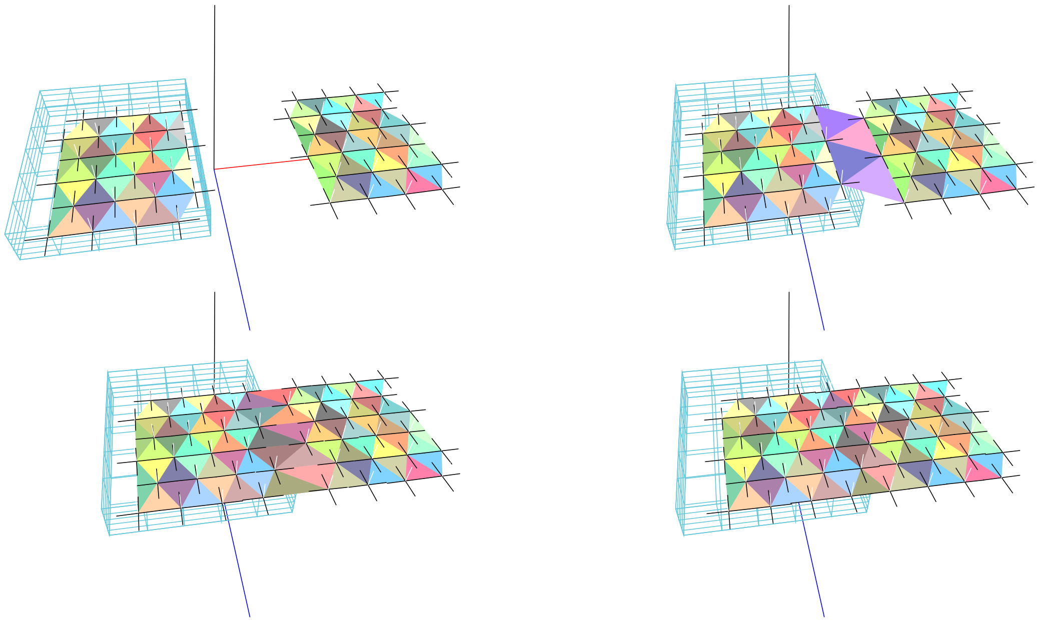


Figure 6: Welding two surfaces together. The two surfaces are brought together through interactive user manipulation, and join to become one seamless surface.

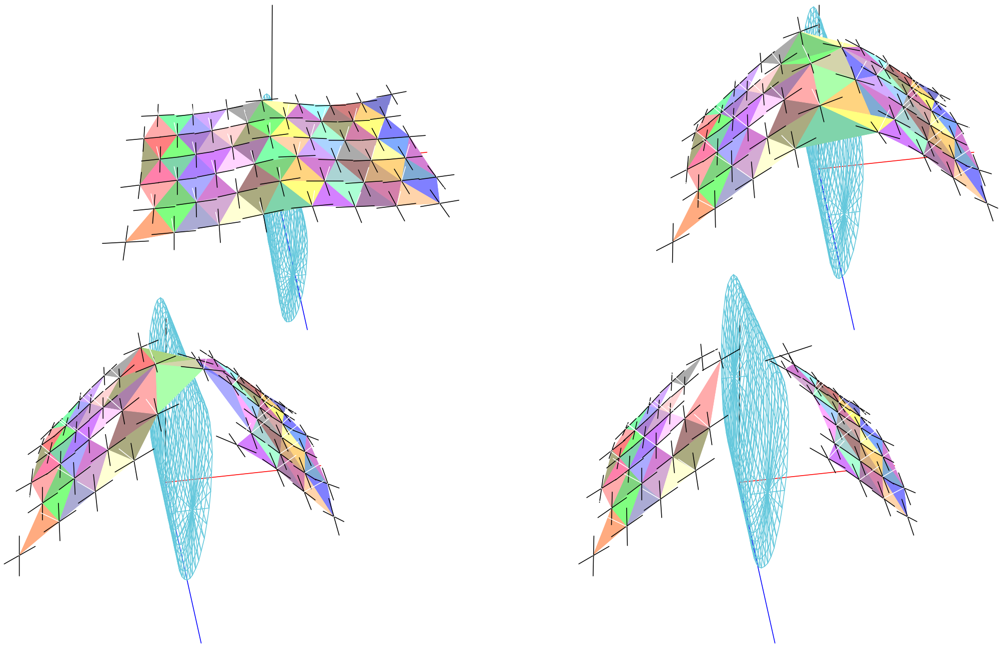


Figure 7: Cutting a surface into two. The movement of the knife edge pushes the particles in the two surfaces apart.

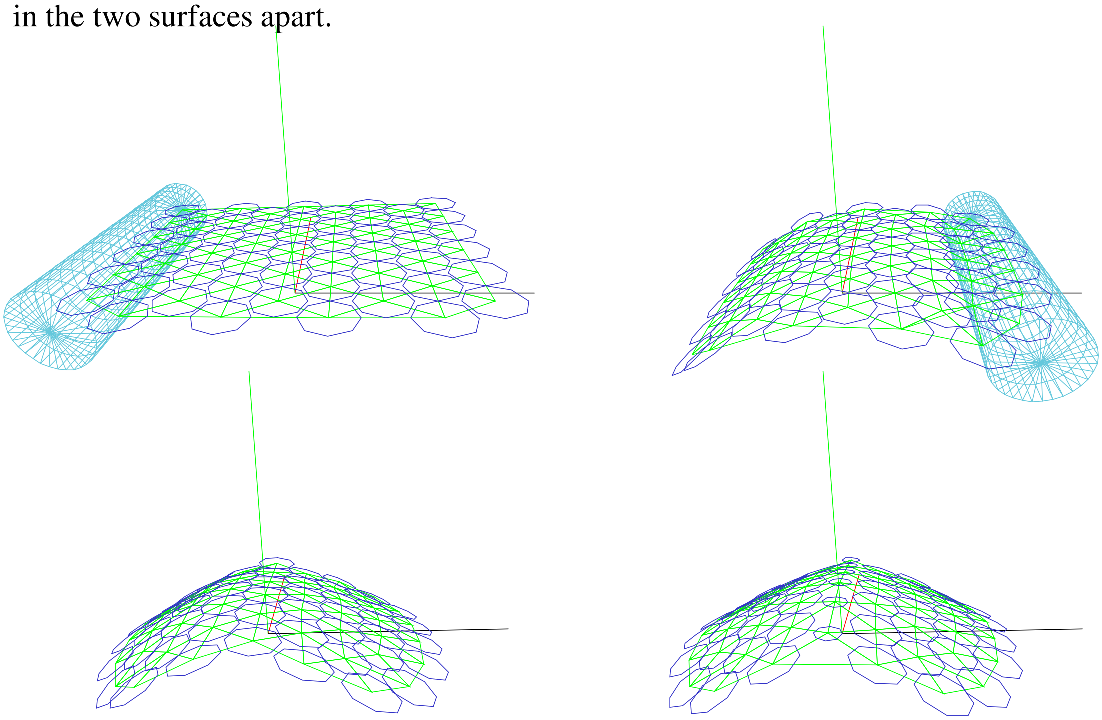


Figure 8: Putting a crease into a surface. The center row of particles is turned into unoriented particles which ignore smoothness forces.

thereby locally disabling surface smoothness forces (co-planarity, etc.) without removing inter-particle spacing interactions (Figure 8). The automatic placement of such creases and jump discontinuities during surface interpolation is a problem that has been extensively studied in the computer vision literature [Terzopoulos, 1988].

7 Particle creation and 3-D interpolation

Our particle-based modeling system can be used to shape a wide variety of surfaces by interactively creating and manipulating particles. This modeling system becomes even more flexible and powerful if surface extension occurs automatically or semi-automatically. For example, we would like to stretch a surface and have new particles appear in the elongated region, or to fill small gaps in the surface, or extend the surface at its edges. Another useful capability would be a system which can fit a surface to an arbitrary collection of 3-D points. Below, we describe how our system can be extended to generate such behaviors.

The basic components of our particle-based surface extension algorithm are two heuristic rules controlling the addition of new particles. These rules are based on the assumption that the particles on the surface are in a near-equilibrium configuration with respect to the flatness, bending, and inter-particle spacing potentials.

The first (*stretching*) rule checks to see if two neighboring particles have a large enough opening between them to add a new particle. If two particles are separated by a distance d such that $d_{\min} \leq d \leq d_{\max}$, we create a candidate particle at the midpoint and check if there are no other particles within $1/2\, d_{\min}$. The particle's position and orientation are obtained from a weighted average of the positions and orientations of nearby particles. Typically $d_{\min} \approx 2.0\, r_0$ and $d_{\max} \approx 2.5\, r_0$, where r_0 is the natural inter-particle spacing. An example of this stretching rule in action is shown in Figure 9, where a ball pushing against a sheet stretches it to the point where new particles are added.

The second (*growing*) rule allows particles to be added in all directions with respect to a particle's local x-y plane. The rule is generalized to allow a minimum and maximum number of neighbors and to limit growth in regions of few neighboring particles, such as at the edge of a surface. The rule counts the number of immediate neighbors n_N to see if it falls within a valid range $n_{\min} \leq n_N \leq n_{\max}$. It also computes the angles between successive neighbors $\Delta\theta_i = \theta_{i+1} - \theta_i$ using the particle's local coordinate frame, and checks if these fall within a suitable range $\theta_{\min} \leq \Delta\theta_i \leq \theta_{\max}$. If these conditions are met, one or more particles are created in the gap. In general, a sheet at equilibrium will have interior particles with six neighbors spaced 60° apart while edge particles will have four neighbors with one pair of neighbors 180° apart.

With these two rules, we can automatically build a surface from collections of 3-D points. We create particles at each sample location and fix their positions and orientations. We then start filling in gaps by growing particles away from isolated points and edges. After a rough surface approximation is complete we can release the original sampled particles to smooth the final surface thereby eliminating excessive noise. If the set of data points is reasonably distributed, this approach will result in a smooth continuous closed surface (Figure 10). The fitted surface is not limited to a particular topology, unlike previous 3-D surface fitting models such as [Terzopoulos *et al.*, 1987b; Miller *et al.*, 1991].

We can also fit surfaces to data that does not originate from closed surfaces, such as stereo range data [Barnard and Fischler, 1982; Fua and Sander, 1992; Szeliski, 1991]. Simply growing particles away from the sample points poses several problems. For example, if we allow growth in all directions, the surface may grow indefinitely at the edges, whereas if we limit the growth at edges, we may not be able to fill in certain gaps. Instead, we apply the stretching heuristic to effectively interpolate the surface between the sample points (Figure 11). When the surface being reconstructed has holes or gaps, we can control the size of gaps that are filled in by limiting the search range. This is evident in Figure 11, where the cheek and neck regions have few samples and were therefore not reconstructed. We could have easily filled in these regions by using a larger search range.

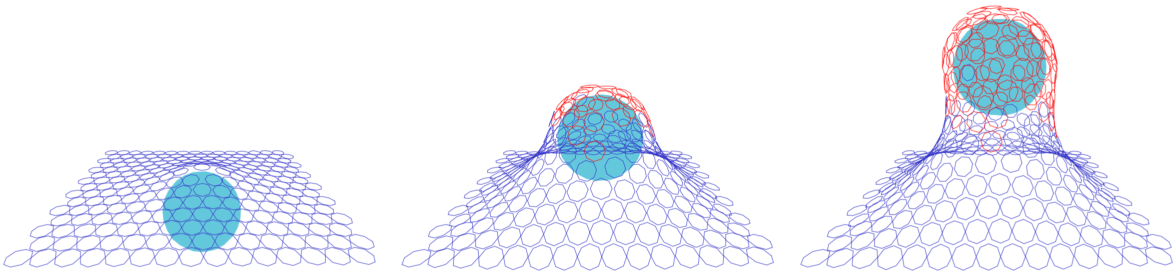


Figure 9: Particle creation during stretching. As the ball pushes up through the sheet, new particles are created in the gaps between pairs of particles.

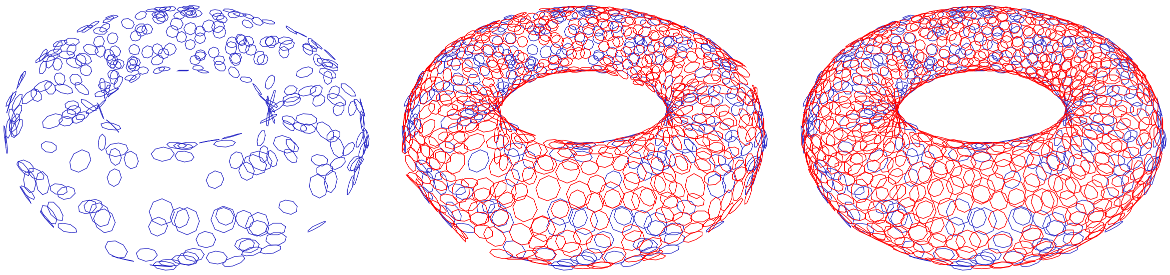


Figure 10: Surface interpolation through a collection of 3-D points. The surface extends outward from the seed points until it fills in the gaps and forms a complete surface.

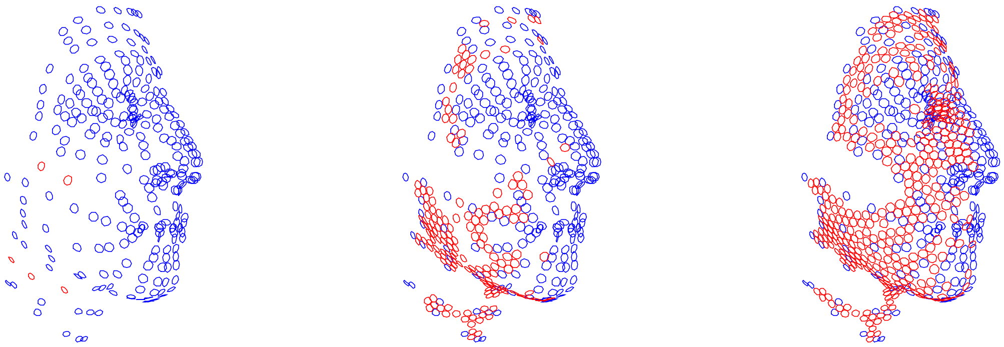


Figure 11: Interpolation of an open surface through a collection of 3-D points. Particles are added between control points until all gaps less than a specified size are filled in. Increasing the range would allow the sparse areas of the cheek and neck to filled in.

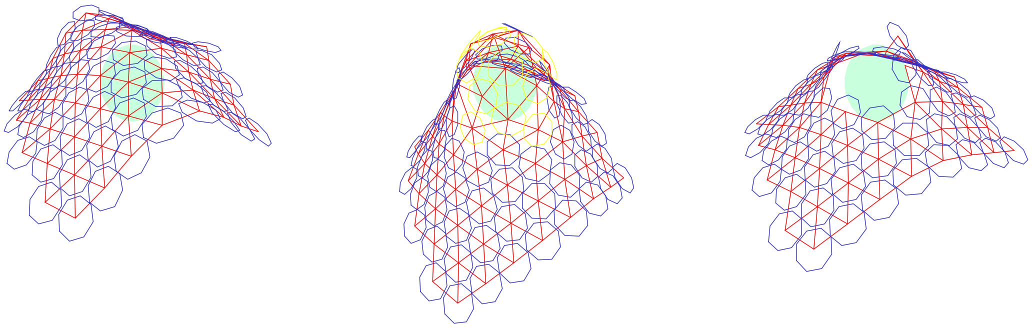


Figure 12: Varying the surface characteristics to change its behavior: (a) cloth draping, (b) plastic deformation, (c) tearing.

8 Geometric Modeling Applications

The particle-based surface models we have presented can be used in a wide range of geometric modeling and animation applications. These include applications which have been previously demonstrated with physically-based deformable surface models, such as cloth draping [Weil, 1986; Terzopoulos *et al.*, 1987b; Breen *et al.*, 1991] (Figure 12a), plastic surface deformations [Terzopoulos and Fleischer, 1988b] (Figure 12b), and tearing [Terzopoulos and Fleischer, 1988b] (Figure 12c).

Using our surface model as an interactive design tool we can spray collections of points into space to form elastic sheets, shape them under interactive user control, and then freeze them into the desired final configuration. We can create any desired topology with this technique. For example, we can form a flat sheet into an object with a stem and then a handle (Figure 13). Forming such surface with traditional spline patches is a difficult problem that requires careful attention to patch continuities [Loop and DeRose, 1990]. To make this example work, we add the concept of *heating* the surface near the tool [Tonnesen, 1991] and only allowing the hot parts of the surface to deform and stretch. Without this modification, the extruded part of the surface has a tendency to "pinch off" similar to how soap bubbles pinch before breaking away. As another example, we can start with a sphere,

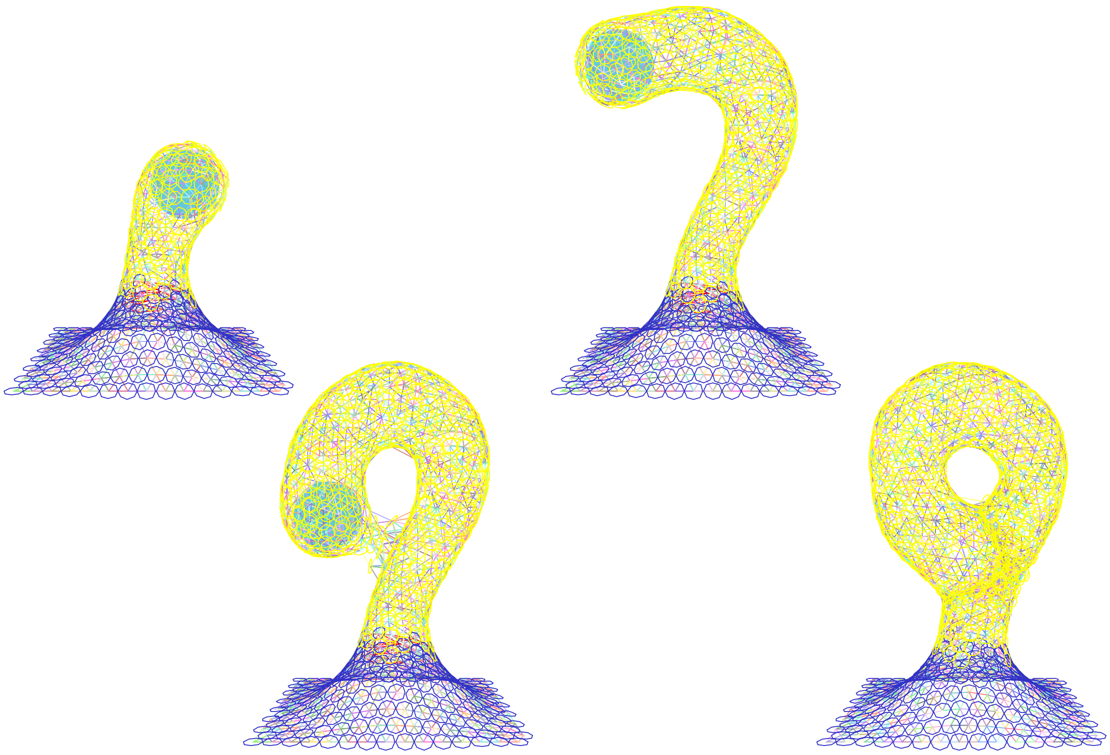


Figure 13: Forming a complex object. The initial surface is deformed upwards and then looped around. The new topology (a handle) is created automatically.

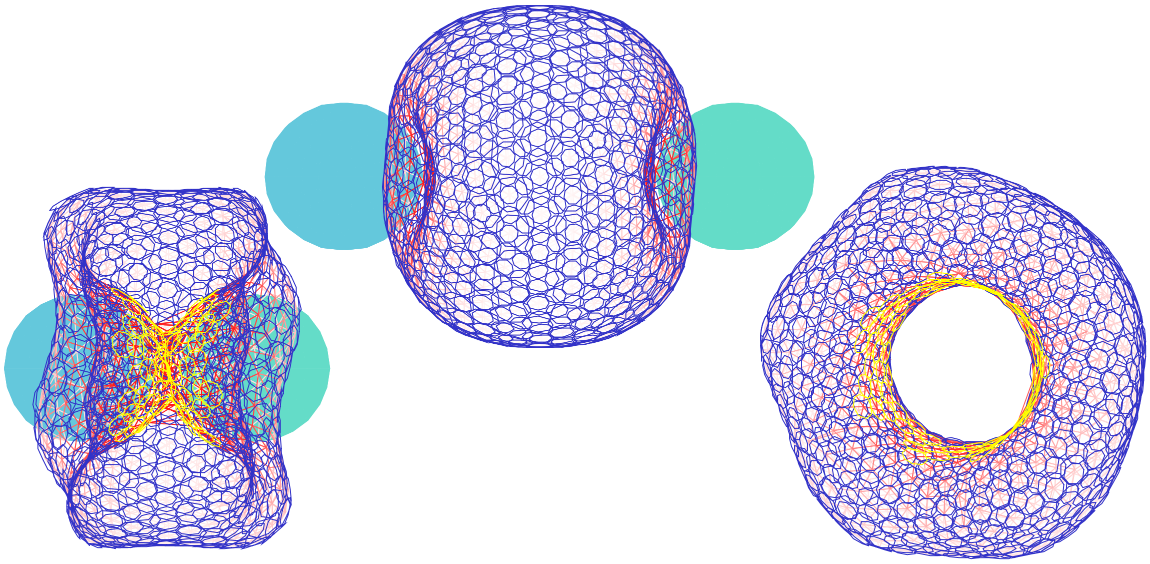


Figure 14: Deformation from sphere to torus using two spherical shaping tools. The final view is from the side, showing the toroidal shape.

and by pushing in the two ends, form it into a torus (Fig 14). New particles are created inside the torus due to stretching during the formation process, and some old sphere particles are deleted when trapped between the two shaping tools.

Another interesting application of our oriented particle systems is the interpolation and extrapolation of sparse 3-D data. This is a difficult problem when the topology or rough shape of the surface to be fitted is unknown. As described in the previous section, oriented particles provide a solution by extending the surface out from known data points. We believe that these techniques will be particularly useful in machine vision applications where it can be used to interpolate sparse position measurements available from stereo or tactile sensing [Szeliski, 1991].

A direct extension of our surface fitting procedure is to add a potential function that induces a torque around the local z axis. This torque can be used to force the x and y axes to align themselves in the directions of minimum and maximum curvature. For example, the potential term (expressed in local coordinates)

$$\phi_{\mathrm{S}}(x, y, z) = x^2 z \, \psi(x, y, z) \tag{14}$$

encourages the x axis to align itself in the direction of smallest curvature (greatest negative curvature). Similarly, the potential term

$$\phi_{\mathrm{S}}(x, y, z) = x^2 z^2 \, \psi(x, y, z) \tag{15}$$

encourages the x axis to align itself in the direction of smallest *absolute* 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 15). The resulting system of oriented particles resembles the collection of interacting Darboux frames used by Sander and Zucker [1990].

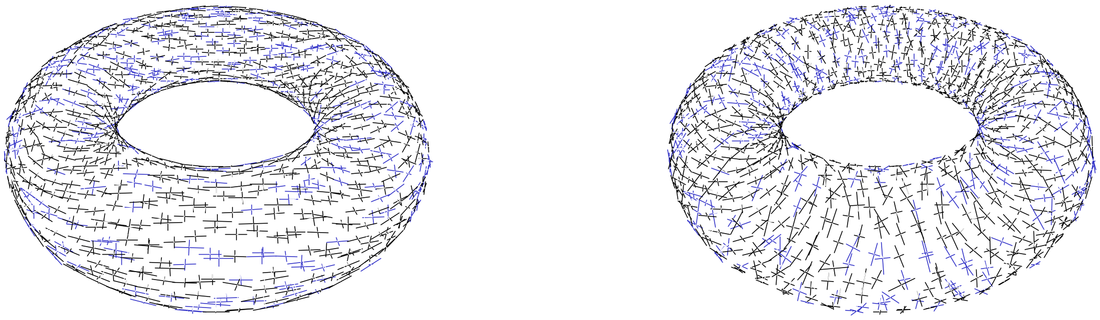


Figure 15: The spin torque potential ϕ_S forces the local coordinate frames to align with the minimum and maximum curvatures of the surface (short and long axes, respectively). The left and right images are before and after the addition of ϕ_S.

9 Discussion

The particle-based surface model we have developed has a number of advantages over traditional spline-based and physically-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.

One 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 torus shaped from a sphere is not circularly symmetric, due to the discretization effects of the relatively small number of particles. This behavior could be remedied by adding additional 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 maximum 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. For example, particle-based surface patches could be added to a constructive solid geometry (CSG) modeling system to perform fileting at part junctions.

In future work, we will apply particle-based surfaces to iso-surfaces in volumetric data sets. When combined with the stretching heuristic for particle creation and an inflation force, this model would behave in a manner similar to the geometrically deformed models (GDM) of [Miller *et al.*, 1991]. We could extend this idea by tracking a volumetric data set through time by deforming the particle surface from one frame to the next.

In another application, we could distribute the particles over the surface of a CAD model and allow the particles to change position and orientation while remaining on the surface of the model, thereby creating a uniform triangulation of the surface. Figure 11 shows how this can be achieved, even without the presence of the CAD model surface to attract the particles. A curvature-dependent adaptive meshing of the surface could also be obtained by locally adjusting the preferred inter-particle spacing. This would be very useful for efficiently rendering parametric surfaces such as NURBS.

10 Conclusion

In this paper, we have developed a particle-based model of deformable surfaces. 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.

Like previous deformable surface models, our new particle-based surfaces can simulate cloth, elastic and plastic films, and other deformable surfaces. The ability to grow new particles gives these model more fluid-like properties which extend the range of interactions. For example, the surfaces can be joined and cut at arbitrary locations. These characteristics make particle-based surfaces a powerful new tool for the interactive construction and modeling of free-form surfaces.

Oriented particles can also be used to automatically fit a surface to sparse 3-D data even when the topology of the surface is unknown. Both open and closed surfaces can be reconstructed, either with or without holes. The reconstructed model can be used as the starting point to interactively create a new shape and then animated within a virtual environment. Thus oriented particle systems provide a convenient interface between surface reconstruction in computer vision, free form modeling in computer graphics, and animation.

References

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

[Barnard and Fischler, 1982] S. T. Barnard and M. A. Fischler. Computational stereo. *Computing Surveys*, 14(4):553–572, December 1982.

[Bartels and Beatty, 1989] R. H. Bartels and J. C. Beatty. A technique for the direct manipulation of spline curves. In *Graphics Interface '89*, pages 33–39, June 1989.

[Bartels *et al.*, 1987] R. H. Bartels, J. C. Beatty, and B. A. Barsky. *An Introduction to Splines for use in Computer Graphics and Geeometric Modeling*. Morgan Kaufmann Publishers, Los Altos, California, 1987.

[Blinn, 1982a] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics (SIGGRAPH'82)*, 16(3):21–29, July 1982.

[Blinn, 1982b] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[Boissonat, 1984] 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.

[Breen *et al.*, 1991] D. E. Breen, D. H. House, and P. H. Getto. A particle-based computational model of cloth draping behavior. In N. M. Patrikalakis, editor, *Scientific Visualization of Physical Phenomena*, pages 113–134, Springer-Verlag, New York, 1991.

[Celniker and Gossard, 1991] 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.

[Coquillart, 1990] S. Coquillart. Extended free-form deformations: A sculpturing tool for 3d geometric modeling. *Computer Graphics (SIGGRAPH'90)*, 24(4):187–196, August 1990.

[Farin, 1990] G. E. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, Boston, Massachusetts, 2nd edition, 1990.

[Fua and Sander, 1992] P. Fua and P. Sander. Reconstructing surfaces from unstructured 3d points. In *Second European Conference on Computer Vision (ECCV'90)*, (submitted) 1992.

[Hersh, 1987] J. Hersh. *Soft Object Extensions to The Clockworks system*. Technical Report TR-87054, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1987.

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

[Kass *et al.*, 1988] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, January 1988.

[Loop and DeRose, 1990] C. Loop and T. DeRose. Generalized B-spline surfaces of arbitrary topology. *Computer Graphics (SIGGRAPH'90)*, 24(4):347–356, August 1990.

[Miller and Pearce, 1989] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. In *SIGGRAPH '89, Course 30 notes: Topics in Physically-based Modeling*, pages R1 – R23, SIGGRAPH, August 1989. Boston, Massachusetts.

[Miller *et al.*, 1991] J. V. Miller, D. E. Breen, W. E. Lorensen, R. M. O'Bara, and M. J. Wozny. Geometrically deformed models: A method of extracting closed geometric models from volume data. *Computer Graphics (SIGGRAPH'91)*, 25(4):217–226, July 1991.

[Platt and Barr, 1988] J. C. Platt and A. H. Barr. Constraint methods for flexible models. *Computer Graphics (SIGGRAPH'88)*, 22(4):279–288, August 1988.

[Press *et al.*, 1988] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988.

[Reeves, 1983] W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions of Graphics*, 2(2):91–108, April 1983.

[Reeves and Blau, 1985] W. T. Reeves and R. Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics*, 19(3):313–322, July 1985.

[Sachs *et al.*, 1991] E. Sachs, A. Roberts, and D. Stoops. 3-draw: A tool for designing 3d shapes. *IEEE Computer Graphics & Applications*, 11(6):18–26, November 1991.

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

[Sander and Zucker, 1990] 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.

[Shoemake, 1985] K. Shoemake. Animating rotation with quaternion curves. *Computer*

Graphics (SIGGRAPH'85), 19(3):245–2540, July 1985.

[Sims, 1990] K. Sims. Particle animation and rendering using data parallel computation. *Computer Graphics (SIGGRAPH'90)*, 24(4):405–413, August 1990.

[Szeliski, 1991] R. Szeliski. Shape from rotation. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'91)*, pages 625–630, IEEE Computer Society Press, Maui, Hawaii, June 1991.

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

[Terzopoulos and Fleischer, 1988a] D. Terzopoulos and K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, December 1988.

[Terzopoulos and Fleischer, 1988b] D. Terzopoulos and K. Fleischer. Modeling inelastic deformations: Visoelasticity, plasticity, fracture. *Computer Graphics (SIGGRAPH'88)*, 22(4):269–278, August 1988.

[Terzopoulos *et al.*, 1989] D. Terzopoulos, J. Platt, and K. Fleischer. From goop to glop: Heating and melting deformable models. In *Proceedings Graphics Interface*, pages 219–226, Graphics Interface, June 1989.

[Terzopoulos *et al.*, 1987a] D. Terzopoulos, A. Witkin, and M. Kass. Symmetry-seeking models and 3D object reconstruction. *International Journal of Computer Vision*, 1(3):211–221, October 1987.

[Terzopoulos *et al.*, 1987b] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics (SIGGRAPH'87)*, 21(4):205–214, July 1987.

[Tonnesen, 1989] D. Tonnesen. *Ray-tracing implicit surfaces resulting from the summation of bounded polynomial functions*. Technical Report TR-89003, Rensselaer Design Research Center, Rensselaer Polytechnic Institute, Troy, New York, 1989.

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

[Weil, 1986] J. Weil. The synthesis of cloth objects. *Computer Graphics (SIGGRAPH'86)*, 20(4):49–54, August 1986.

[Witkin and Welch, 1990] A. Witkin and W. Welch. Fast animation and control of nonrigid structures. *Computer Graphics (SIGGRAPH'90)*, 24(4):243–252, August 1990.

[Wolfram, 1988] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley, Redwood City, California, 1988.

[Wyvill *et al.*, 1986] B. Wyvill, C. McPheeters, and G. Wyvill. Data structures for soft objects. *The Visual Computer*, 2:227–234, 1986.

A Finite element analysis of local deformation energies

To derive the local oriented particle interaction potentials, we analyze the deformation energies of a triangular surface patch defined by three neighboring particles. For this analysis, we assume that the particles are in an equilateral configuration with locations $(0, 0)$, $(h, 0)$ and $(1/2, \sqrt{3}/2)$ in the (x, y) plane. We examine the small-deflection case where the height from the plane, $z = f(x, y)$, describes the local shape of the surface. Both of these assumptions are reasonable for our surfaces, since the Lennard-Jones forces favor locally hexagonal arrangements, and a sufficiently high sampling density will ensure small deflections. For an analysis of the general parametric patch case, see [Celniker and Gossard, 1991].

We use a cubic function for $f(x, y)$ since it can be specified by the heights and gradients at the three corners $\{(z_i, p_i, q_i), i = 0 \ldots 2\}$ and the height z_3 of a "bubble" node in the middle of a triangle. We choose the (x, y) plane to pass through the three particles, which gives us a height of 0 at all three corners.

To compute the deformation energies, we take integrals of squared derivatives over the triangle. For example, we can compute the area of the triangle from

$$A = \int\int \sqrt{1 + f_x^2 + f_y^2}\, dx\, dy \approx \frac{\sqrt{3}}{4}h^2 + \frac{1}{2}\int\int f_x^2 + f_y^2\, dx\, dy.$$

We can compute the average Gaussian curvature from

$$C \approx \frac{1}{2}\int\int f_{xx}^2 + 2f_{xy}^2 + f_{yy}^2\, dx\, dy$$

and the average variation in curvature from

$$V \approx \frac{1}{2}\int\int f_{xxx}^2 + 3f_{xxy}^2 + 3f_{xyy}^2 + f_{yyy}^2\, dx\, dy.$$

These three integrals can be thought of as corresponding to the stretching, bending, and "undulation" energies of the surface. After some algebraic manipulation, which we performed using Mathematica™[Wolfram, 1988], we obtain formulas for the above three equations in terms of the corner gradient values $\{(p_i, q_i)\}$ and the bubble height z_3 (the expressions are quadratic in these variables).

In our oriented particle system, we desire to have interactions only between pairs of particles. Since we are only interested in the energies involving two particles, say the particles which control (p_0, q_0) and (p_1, q_1), we minimize the quadratic energies with respect to the p_2, q_2, and z_3 variables (this results in lower energies than arbitrarily setting these unknown quantities to 0, which would be the effect of ignoring these other terms). To further simplify the energies, we express them in terms of averages and differences of gradients

$$\begin{aligned} p_+ &= (p_0 + p_1)/2 \qquad & q_+ &= (q_0 + q_1)/2 \\ p_- &= (p_0 - p_1)/2 \qquad & q_- &= (q_0 - q_1)/2. \end{aligned}$$

Again, applying the tools in Mathematica™, we obtain

$$V = h^{-2} 6\sqrt{3} p_+^2, \tag{16}$$

$$C = \frac{\sqrt{3}}{2268}(567p_+^2 + 316p_-^2 + 8\sqrt{3}p_-q_+ + 48q_+^2 + 315q_-^2), \tag{17}$$

$$A = h^2\frac{\sqrt{3}}{4}(1 + \frac{6003}{281880}p_+^2 + \cdots). \tag{18}$$

How do we compute these quantities given the state of two particles, i.e., their positions and orientations? We must first write the scalar quantities p_0, p_1, q_0, and q_1 in terms of $\mathbf{n}_i$, $\mathbf{n}_j$ and $\mathbf{r}_{ij}$. We identify $\mathbf{r}_{ij}$ with the x direction in our local plane, and thus compute

$$p_0 \approx -\mathbf{n}_i \cdot \hat{\mathbf{r}}_{ij} \text{ and } p_1 \approx -\mathbf{n}_j \cdot \hat{\mathbf{r}}_{ij}$$

for small values of p_0 and p_1. Choosing the y direction is more difficult if we wish to keep the interactions pairwise, since we cannot use the location of the third point defining the triangle. A simple choice is to choose the local z direction along the average normal vector $(\mathbf{n}_i + \mathbf{n}_j)/2$, which leads to the equations

$$\begin{aligned} q_+ &= -(\frac{\mathbf{n}_i + \mathbf{n}_j}{2}) \cdot \frac{\widehat{\mathbf{n}_i + \mathbf{n}_j}}{2} \times \hat{\mathbf{r}}_{ij} = 0, \\ q_- &= -(\frac{\mathbf{n}_i - \mathbf{n}_j}{2}) \cdot \frac{\widehat{\mathbf{n}_i + \mathbf{n}_j}}{2} \times \hat{\mathbf{r}}_{ij}, \\ p_-^2 + q_-^2 &\approx \frac{1}{4}\|\mathbf{n}_i - \mathbf{n}_j\|^2. \end{aligned}$$

We are now in a position to relate the finite element based measures for curvature and variation in curvature to the co-planarity, co-normality, and co-circularity measures. The variation in curvature V (17) corresponds directly to the co-circularity ϕ_{C} (8). The curvature itself C (18) can be written as a sum of the co-circularity potential and the co-normality potential ϕ_{N} (7). The co-planarity potential is therefore not needed to write a curvature-based energy measure. It is useful, however, when used in isolation, since it corresponds to terms of the form

$$p_0^2 + p_1^2 \propto p_+^2 + p_-^2.$$

While the area-based measure A (18) is too complicated to warrant direct implementation, finite rest area behavior is simulated by the Lennard-Jones interaction potential ϕ_{LJ}.

B Computation of internal forces

To compute the internal inter-particle forces and torques, we compute the variation of inter-particle potentials with respect to particle positions and orientations. We can compute these forces and torques using the equations

$$\begin{aligned} \mathbf{f} &= -\nabla_{\mathbf{p}}\phi \quad \text{and} \quad \nabla_{\mathbf{p}}(\mathbf{p}\cdot\mathbf{v}) = \mathbf{v} \\ \boldsymbol{\tau} &= -\nabla_{\boldsymbol{\omega}}\phi \quad \text{and} \quad \nabla_{\boldsymbol{\omega}}(\mathbf{n}\cdot\mathbf{v}) = \mathbf{n}\times\mathbf{v} \end{aligned}$$

where $\boldsymbol{\omega}$ is the incremental change in orientation $\mathbf{R}$, i.e., $\mathbf{n} = \mathbf{n}\times\boldsymbol{\omega}$.

Applying these equations to the various internal potentials, we obtain

$$\begin{aligned}
\mathbf{f}_{\mathrm{LJ}}(\mathbf{r}_{ij}) &= -\hat{\mathbf{r}}_{ij}\,\phi'_{\mathrm{LJ}}(\|\mathbf{r}_{ij}\|) \\
\mathbf{f}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{r}_{ij}) &= -\mathbf{n}_i(\mathbf{n}_i\cdot\mathbf{r}_{ij})\,\psi(\|\mathbf{r}_{ij}\|) - \hat{\mathbf{r}}_{ij}(\mathbf{n}_i\cdot\mathbf{r}_{ij})^2\,\psi'(\|\mathbf{r}_{ij}\|) \\
\boldsymbol{\tau}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{r}_{ij}) &= \mathbf{r}_{ij}\times\mathbf{n}_i(\mathbf{n}_i\cdot\mathbf{r}_{ij})\,\psi(\|\mathbf{r}_{ij}\|) = \mathbf{r}_{ij}\times\mathbf{f}_{\mathrm{P}} \\
\mathbf{f}_{\mathrm{N}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) &= -\hat{\mathbf{r}}_{ij}\|\mathbf{n}_i-\mathbf{n}_j\|^2\,\psi'(\|\mathbf{r}_{ij}\|) \\
\boldsymbol{\tau}_{\mathrm{N}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) &= (\mathbf{n}_i-\mathbf{n}_j)\times\mathbf{n}_i\,\psi(\|\mathbf{r}_{ij}\|) = \mathbf{n}_i\times\mathbf{n}_j\,\psi(\|\mathbf{r}_{ij}\|) \\
\mathbf{f}_{\mathrm{C}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) &= -\mathbf{n}_i((\mathbf{n}_i+\mathbf{n}_j)\cdot\mathbf{r}_{ij})\,\psi(\|\mathbf{r}_{ij}\|) - \hat{\mathbf{r}}_{ij}((\mathbf{n}_i+\mathbf{n}_j)\cdot\mathbf{r}_{ij})^2\,\psi'(\|\mathbf{r}_{ij}\|) \\
\boldsymbol{\tau}_{\mathrm{C}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) &= \mathbf{r}_{ij}\times\mathbf{n}_i((\mathbf{n}_i+\mathbf{n}_j)\cdot\mathbf{r}_{ij})\,\psi(\|\mathbf{r}_{ij}\|) = \mathbf{r}_{ij}\times\mathbf{f}_{\mathrm{C}}
\end{aligned}$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector along $\mathbf{r}_{ij}$. These forces have the following simple physical interpretations.

The co-planarity potential gives rise to a force parallel to the particle normal and proportional to the distance between the neighboring particle and the local tangent plane. The second term in the force, which can often be ignored, arises from the gradient of the spatial weighting function. The cross product of this force with the inter-particle vector produces a torque on the particle. The co-normality potential produces a torque proportional to the cross-product of the two particle normals, which acts to lign up the normals. The co-circularity force is similar to the co-planarity force, except that the local tangent plane is defined from the average of the two normal vectors.

To compute the total inter-particle force and torque from all three potentials, we use the formulas

$$\begin{aligned}
\mathbf{f}_i &= \sum_{j\in\mathcal{N}_i} 2\alpha_{\mathrm{LJ}}\mathbf{f}_{\mathrm{LJ}}(\mathbf{r}_{ij}) + \alpha_{\mathrm{P}}(\mathbf{f}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{r}_{ij}) - \mathbf{f}_{\mathrm{P}}(\mathbf{n}_j,\mathbf{r}_{ji})) \\
&\quad + 2\alpha_{\mathrm{N}}\mathbf{f}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) + 2\alpha_{\mathrm{C}}\mathbf{f}_{\mathrm{C}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) \qquad (19)\\
\boldsymbol{\tau}_i &= \sum_{j\in\mathcal{N}_i} \alpha_{\mathrm{P}}\boldsymbol{\tau}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{r}_{ij}) + 2\alpha_{\mathrm{N}}\boldsymbol{\tau}_{\mathrm{P}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) + 2\alpha_{\mathrm{C}}\boldsymbol{\tau}_{\mathrm{C}}(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) \qquad (20)
\end{aligned}$$

Note that most forces and torques are doubled, i.e., actions generate opposite reactions. The exception to this is $\mathbf{f}_{\mathrm{P}}$ and $\boldsymbol{\tau}_{\mathrm{P}}$, which arise from an asymmetric potential function. This can easily be changed by defining a new potential

$$\phi_{\mathrm{P}}^*(\mathbf{n}_i,\mathbf{n}_j,\mathbf{r}_{ij}) = \phi_{\mathrm{P}}(\mathbf{n}_i,\mathbf{r}_{ij}) + \phi_{\mathrm{P}}(\mathbf{n}_j,\mathbf{r}_{ij}).$$