Surface Modeling with Oriented Particle Systems

Richard Szeliski[†] **and David Tonnesen**[‡]

[†]Digital Equipment Corp., Cambridge Research Lab, One Kendall Square, Bldg. 700, Cambridge, MA 02139
[‡]Dept. of Computer Science, University of Toronto, Toronto, Canada M5S 1A4

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: Surface interpolation, particle systems, physically-based modeling, oriented particles, self-organizing systems, simulation.

CR Categories and Subject Descriptors: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling — *Curve, surface, solid, and object representations*; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism — *Animation.*

1 Introduction

The modeling of free-form surfaces is one of the central issues of computer graphics. Spline models [3, 8] and deformable surface models [25] 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 [18]. 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 [16] and Sims [21] 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 [12, 26, 29]. In these models, which have spherically symmetric potential fields, particles arrange themselves into volumes rather than surfaces.

In this paper, we develop *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 [17]. 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 reparameterization 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, as well as particle systems and 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 dy-

ACM-0-89791-479-1/92/007/0185 $01.50

namics (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.

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 [3, 8] model an object's surface as a collection of piecewise-polynomial patches, with appropriate continuity constraints 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

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

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 [2].

Elastically deformable surface models [25] 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.* [25] use weighted combinations of different tensor (stretching and bending) measures to define a simplified deformation energy which controls the elastic restoring forces for the surface. 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 [26].

Physically-based surface models have been used to model a wide variety of materials, including cloth [30, 6], membranes [25], and paper [24]. Viscoelasticity, plasticity, and fracture have been incorporated to widen the range of modeled phenomena [24].

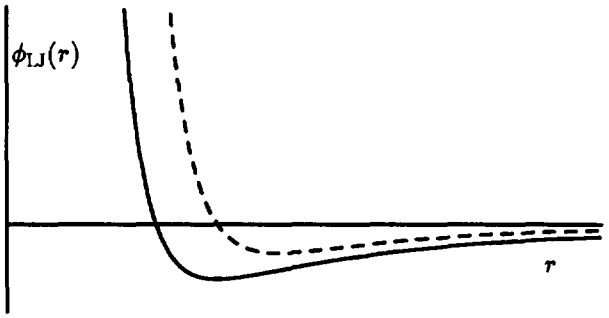

Figure 1: Lennard-Jones type function, $\phi_{LJ}(r) = B/r^n - A/r^m$. The solid line shows the potential function $\phi_{LJ}(r)$, and the dashed line shows the force function $f(r) = -\frac{d}{dr}\phi_{LJ}(r)$.

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 [27]. For spline models, this means discretizing the surface into a collection of patches with appropriate continuity conditions, which is generally a difficult problem [11]. 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 case, defining the model topology in advance remains a tedious process. Furthermore, it severely limits the flexibility of a given surface model.

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 [16] and waterfalls [21]. Interacting (oriented) particles have been used to simulate flocks of "boids" [17].

Ideas from molecular dynamics have been used to develop models of deformable materials using collections of interacting particles [26, 12, 29]. 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

$$\mathbf{f}_{ij} = -\nabla_{\mathbf{r}}\phi_{LJ}(||\mathbf{r}_{ij}||), \qquad (2)$$

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

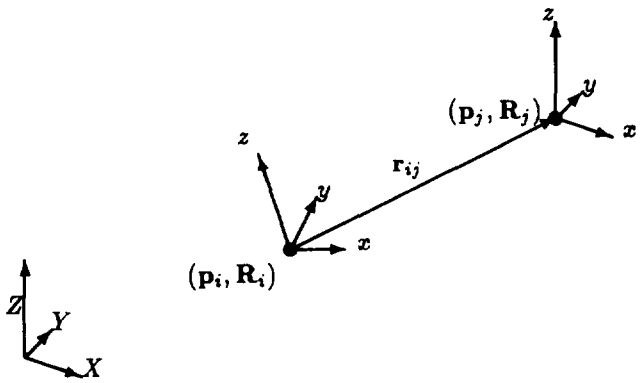

Figure 2: Global and local coordinate frames. The global interparticle distance $\mathbf{r}_{ij}$ is computed from the global coordinates $\mathbf{p}_i$ and $\mathbf{p}_j$ of particles i and j. The local distance $\mathbf{d}_{ij}$ is computed from $\mathbf{r}_{ij}$ and the rotation matrix $\mathbf{R}_i$.

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 [29]. When external forces become larger or internal particle forces smaller, the behavior resembles that of viscous fluids [26, 12]. More sophisticated models of molecular dynamics are used in simulations of physics and chemistry [10]; 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 [23].

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 $\mathbf{n}_i$.

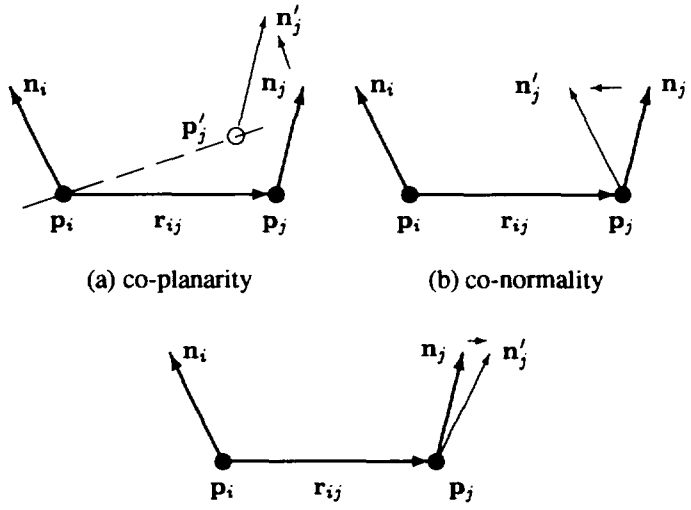

Figure 3: The three oriented particle interaction potentials. The open circles and thin arrows indicate a possible new position or orientation for the second particle which would lead to a null potential.

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

$$\phi_P(\mathbf{n}_i, \mathbf{r}_{ij}) = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \psi(\|\mathbf{r}_{ij}\|), \qquad (3)$$

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

$$\phi_N(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \psi(\|\mathbf{r}_{ij}\|), \qquad (4)$$

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

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

$$\phi_C(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) = ((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij})^2 \psi(\|\mathbf{r}_{ij}\|) \qquad (5)$$

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

$$\mathbf{d}_{ij} = \mathbf{R}_i^{-1} \mathbf{r}_{ij} = \mathbf{R}_i^{-1} (\mathbf{p}_j - \mathbf{p}_i), \qquad (6)$$

which gives the coordinates of particle j in particle i's local coordinate frame. This not only simplifies certain potential equations such as (3), 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

$$\psi(x,y,z) = K\exp\left(-\frac{x^2+y^2}{2a^2} - \frac{z^2}{2b^2}\right) \qquad (7)$$

with $b \leq a$.

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

$$\begin{aligned} E_{ij} = {} & \alpha_{\text{LJ}}\phi_{\text{LJ}}(\|\mathbf{r}_{ij}\|) + \alpha_{\text{P}}\phi_{\text{P}}(\mathbf{n}_i, \mathbf{r}_{ij}) \\ & + \alpha_{\text{N}}\phi_{\text{N}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) + \alpha_{\text{C}}\phi_{\text{C}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}). \end{aligned} \qquad (8)$$

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. The total internal energy of the system E_{int} is computed by summing the inter-particle energies

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

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 A. These forces and torques can be summed over all interacting particles to obtain

$$\mathbf{f}_i = \sum_{j\in\mathcal{N}_i} \mathbf{f}_{ij} + \mathbf{f}_{\text{ext}}(\mathbf{p}_i) - \beta_0\mathbf{v}_i, \qquad (9)$$

$$\boldsymbol{\tau}_i = \sum_{j\in\mathcal{N}_i} \boldsymbol{\tau}_{ij} - \beta_1\boldsymbol{\omega}_i, \qquad (10)$$

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 $\mathbf{f}_{\text{ext}}$, 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

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

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

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

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 [20].

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 [15] to advance the current velocity and position over the time step. More sophisticated numerical integration techniques such as Runge-Kutta [15] or semi-implicit methods [25] could also be used, and would result in better convergence and larger timesteps, at the expense of a more complicated implementation.

4.2 Controlling Complexity

The straightforward evaluation of (9) and (10) 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 [1]. In our work, we use a *k-d tree* [19] 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 [16, 21] and iso-surfaces or "blobbies" [4, 28] 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

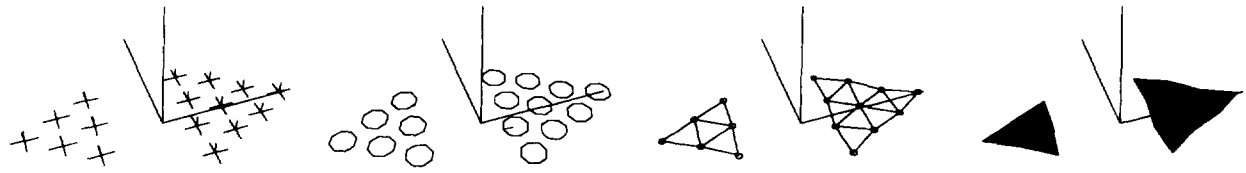

Figure 4: Rendering techniques for particle-based surfaces: (a) axes, (b) discs, (c) wireframe triangulation (d) flat-shaded triangulation.

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 interpolated at each triangle (since we know the normals at each corner).

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 [5]. 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.

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 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 [18] would be of obvious benefit.

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 [14], 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 [7].

Using these tools, particle-based surfaces can be "cold welded" together by abutting their edges (Figure 5). 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 6). Here, we use the "heat" of the cutting tool to weaken the inter-particle bonds [29]. Or we can "crease" a surface by designating a line of particles to be *unoriented*, thereby locally disabling surface smoothness forces (co-planarity, etc.) without removing inter-particle spacing interactions (Figure 7).

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 $1/2\, d_{min}$. Typically $d_{min} \approx 2.0\, r_0$ and $d_{max} \approx 2.5\, r_0$, where

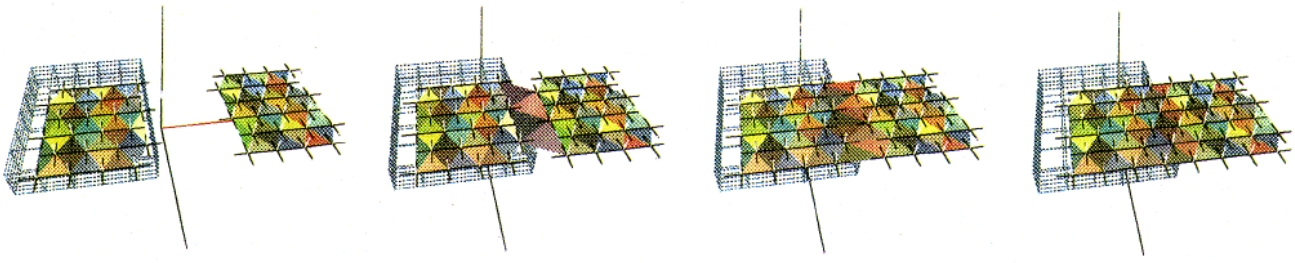

Figure 5: Welding two surfaces together. The two surfaces are brought together through interactive user manipulation, and join to become one seamless surface.

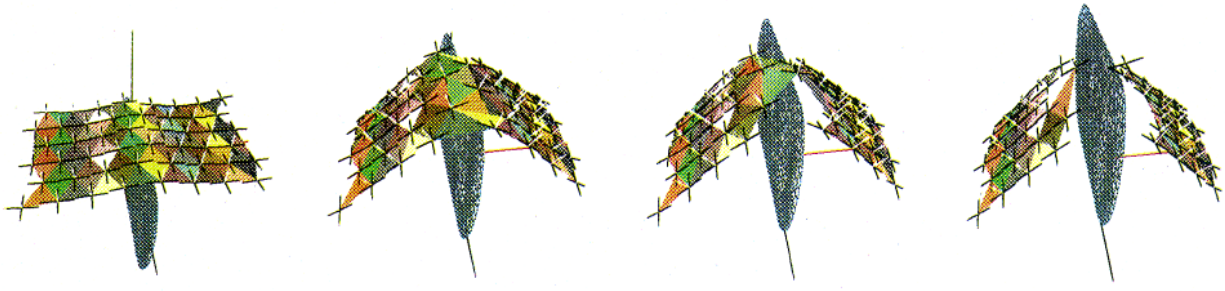

Figure 6: Cutting a surface into two. The movement of the knife edge pushes the particles in the two surfaces apart.

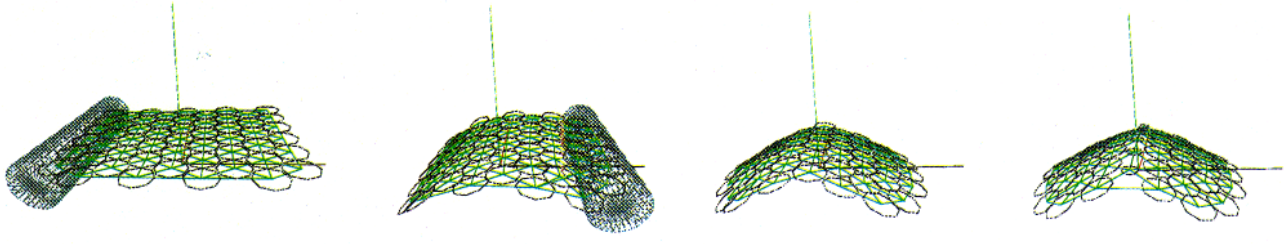

Figure 7: Putting a crease into a surface. The center row of particles is turned into unoriented particles which ignore smoothness forces.

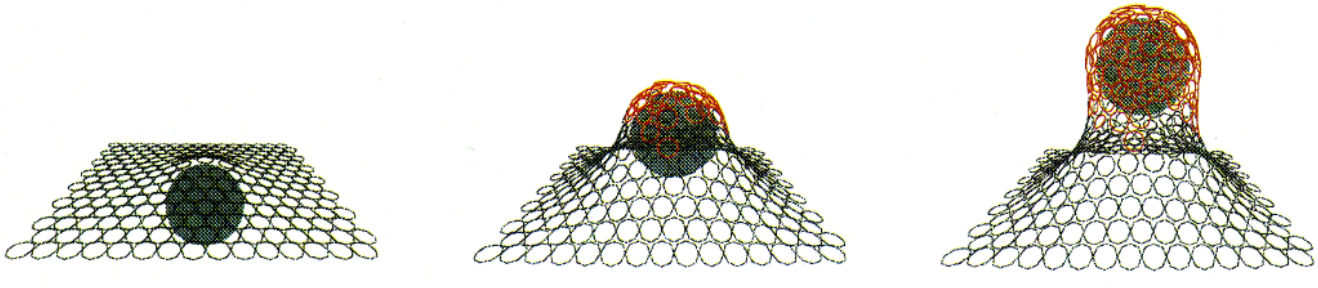

Figure 8: Particle creation during stretching. As the ball pushes up through the sheet, new particles are created in the gaps between pairs of particles.

r_0 is the natural inter-particle spacing. An example of this stretching rule in action is shown in Figure 8, 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 9). The fitted surface is not limited to a particular topology, unlike previous 3-D surface fitting models such as [25, 13].

We can also fit surfaces to data that does not originate from closed surfaces, such as stereo range data [9, 22]. 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 10). 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 10, 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.

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 [30, 25, 6], plastic surface deformations [24], and tearing [24].

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 11). Forming such surface with traditional spline patches is a difficult problem that requires careful attention to patch continuities [11]. To make this example work, we add the concept of *heating* the surface near the tool [29] 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, and by pushing in the two ends, form it into a torus (Fig 12). 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 [22].

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 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.

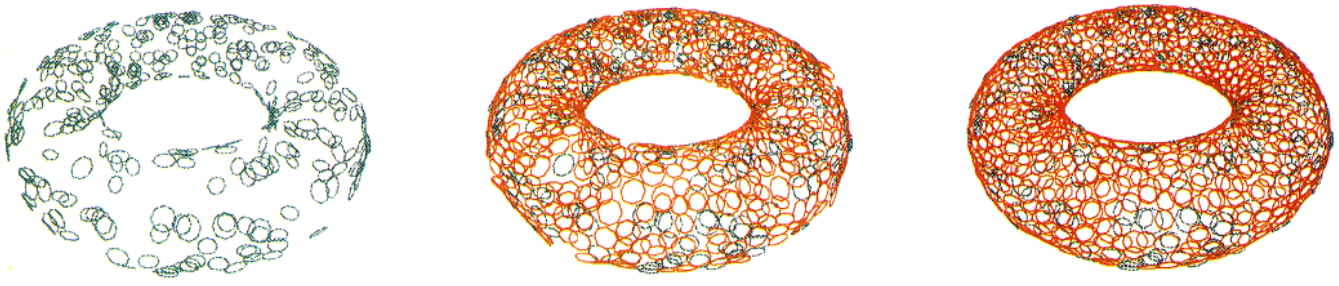

Figure 9: Surface interpolation through a collection of 3-D points. The surface extends outward from the seed points until it fills in the gaps and forms a complete surface.

Figure 10: 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.

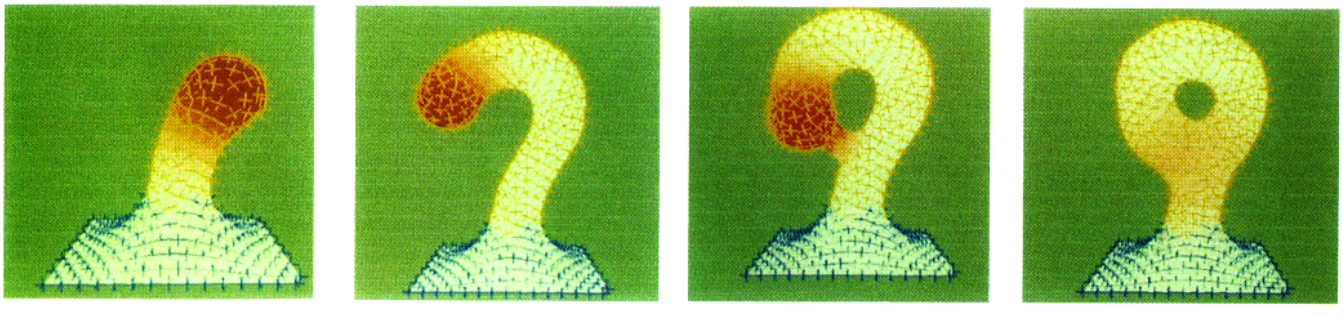

Figure 11: Forming a complex object. The initial surface is deformed upwards and then looped around. The new topology (a handle) is created automatically.

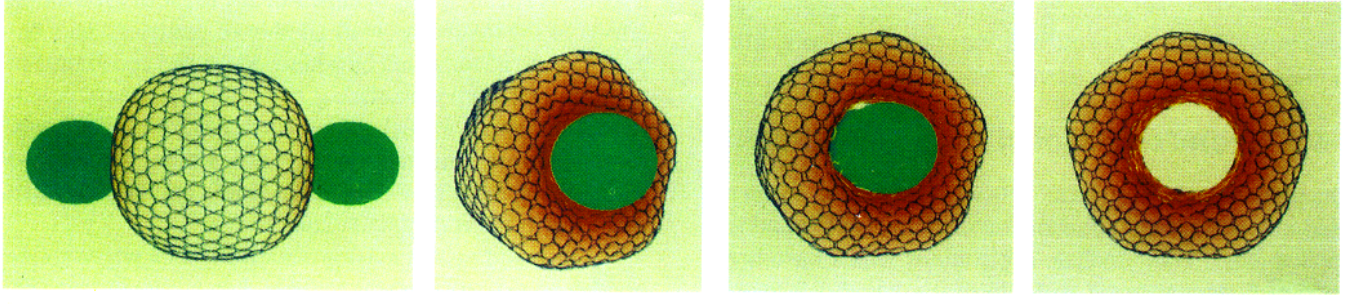

Figure 12: Deformation from sphere to torus using two spherical shaping tools. The final view is from the side, showing the toroidal shape.

In future work, we plan to 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 [13]. 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 10 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. A potential drawback of our technique is the lack of precise control over the mathematical form of the surface, which may be important in engineering applications.

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

[1] Appel, Andrew. An Efficient Algorithm for Many-body Simulations. *SIAM J. Sci. Stat. Comput.*, 6(1), 1985.

[2] Bartels, Richard. H. and Beatty, John. C. A Technique for the Direct Manipulation of Spline Curves. In *Graphics Interface '89*, pages 33–39, June 1989.

[3] Bartels, Richard. H., Beatty, John. C., and Barsky, Brian. A. *An Introduction to Splines for use in Computer Graphics and Geeometric Modeling*. Morgan Kaufmann Publishers, Los Altos, California, 1987.

[4] Blinn, James F. A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

[5] Boissonat, J.-D. 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.

[6] Breen, David E., House, Donald H., and Getto, Phillip H. A Particle-Based Computational Model of Cloth Draping Behavior. In Patrikalakis, N. M., editor, *Scientific Visualization of Physical Phenomena*, pages 113–134. Springer-Verlag, New York, 1991.

[7] Coquillart, Sabine. Extended Free-Form Deformations: A Sculpturing Tool for 3D Geometric Modeling. *Computer Graphics (SIGGRAPH'90)*, 24(4):187–196, August 1990.

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

[9] Fua, Pascal and Sander, Peter. Reconstructing Surfaces from Unstructured 3D Points. In *Second European Conference on Computer Vision (ECCV'92)*, Sta. Margherita, Italy, May 1992. Springer-Verlag.

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

[11] Loop, Charles and DeRose, Tony. Generalized B-spline Surfaces of Arbitrary Topology. *Computer Graphics (SIGGRAPH'90)*, 24(4):347–356, August 1990.

[12] Miller, Gavin and Pearce, Andrew. 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.

[13] Miller, James V., Breen, David E., Lorensen, William E., O'Bara, Robert M., and Wozny, Michael J. Geometrically Deformed Models: A Method of Extracting Closed Geometric Models from Volume Data. *Computer Graphics (SIGGRAPH'91)*, 25(4):217–226, July 1991.

[14] Platt, John C. and Barr, Alan H. Constraint Methods for Flexible Models. *Computer Graphics (SIGGRAPH'88)*, 22(4):279–288, August 1988.

[15] Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, England, 1988.

[16] Reeves, William. T. Particle Systems—A Technique for Modeling a Class of Fuzzy Objects. *ACM Transactions of Graphics*, 2(2):91–108, April 1983.

[17] Reynolds, Craig. W. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics (SIGGRAPH'87)*, 21(4):25–34, July 1987.

[18] Sachs, Emanuel, Roberts, Andrew, and Stoops, David. 3-Draw: A Tool for Designing 3D Shapes. *IEEE Computer Graphics & Applications*, 11(6):18–26, November 1991.

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

[20] Shoemake, Ken. Animating Rotation with Quaternion Curves. *Computer Graphics (SIGGRAPH'85)*, 19(3):245–2540, July 1985.

[21] Sims, Karl. Particle Animation and Rendering Using Data Parallel Computation. *Computer Graphics (SIGGRAPH'90)*, 24(4):405–413, August 1990.

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

[23] Szeliski, Richard and Tonnesen, David. Surface Modeling with Oriented Particle Systems. Technical Report 91/14, Digital Equipment Corporation, Cambridge Research Lab, December 1991.

[24] Terzopoulos, Demetri and Fleischer, Kurt. Modeling Inelastic Deformations: Visoelasticity, Plasticity, Fracture. *Computer Graphics (SIGGRAPH'88)*, 22(4):269–278, August 1988.

[25] Terzopoulos, Demetri, Platt, John, Barr, Alan, and Fleischer, Kurt. Elastically deformable models. *Computer Graphics (SIGGRAPH'87)*, 21(4):205–214, July 1987.

[26] Terzopoulos, Demetri, Platt, John, and Fleischer, Kurt. From Goop to Glop: Heating and Melting Deformable Models. In *Proceedings Graphics Interface*, pages 219–226. Graphics Interface, June 1989.

[27] Terzopoulos, Demetri, Witkin, Andrew, and Kass, Michael. Symmetry-Seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1(3):211–221, October 1987.

[28] Tonnesen, David. 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.

[29] Tonnesen, David. Modeling Liquids and Solids using Thermal Particles. In *Graphics Interface '91*, pages 255–262, 1991.

[30] Weil, Jerry. The Synthesis of Cloth Objects. *Computer Graphics (SIGGRAPH'86)*, 20(4):49–54, August 1986.

A 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., $\dot{\mathbf{n}} = \mathbf{n}\times\boldsymbol{\omega}$.

Applying these equations to the four internal potentials, we obtain

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

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

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