Chapter 8

Applications

8.1 Computer Assisted Animation

For computer assisted animation, we desire a model that can provide a wide range of dynamic properties and the ability to interact with an environment. Our particle systems can simulate a range of behavior in both the volume and surface models. We simulate solid to fluid behavior, infinitely stretchable material, and materials that rip and tear. Once created, the model can be influenced by a larger environment modeled for the animation. We now show examples of various physical behavior.

The rigid behavior of a solid is shown in Figure 8.5. The model is being influenced by gravity and the floor plane. Figure 8.4 shows the same model as a flexible solid. Flexible surface models are also shown in Figures 8.1(a) and 8.2. A semi-rigid model is shown in the first six frames of Figure 8.3 where a solid beam of particles exhibit rigid body motion with slight deformations upon colliding with a spherical object and the floor plane. Fluid like behavior is exhibited when this object then melts in the final frames of Figure 8.3. Tearing behavior is exhibited in Figure 8.1(c) when the force of gravity pulling a sheet over a sphere exceeds the inter-particle binding forces. The same model exhibits stretching behavior when our stretching heuristic is enabled (Figure 8.1(b)). In all of the examples, the models are interacting with the external forces of gravity and collisions, and in one case objects of a different temperature.

8.2 Free Form Modeling of Surfaces

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. For example, we require the basic abilities to extend existing surfaces, to split along arbitrary boundaries, or to join several surfaces together without specifying exact connectivity. Spatially coupled particles provide such features.

There are numerous modeling paradigms that can be employed. We can "mill" a solid block of material into a given shape by deleting all particles which lie outside of a given implicit surface definition. From a geometric surface description we can

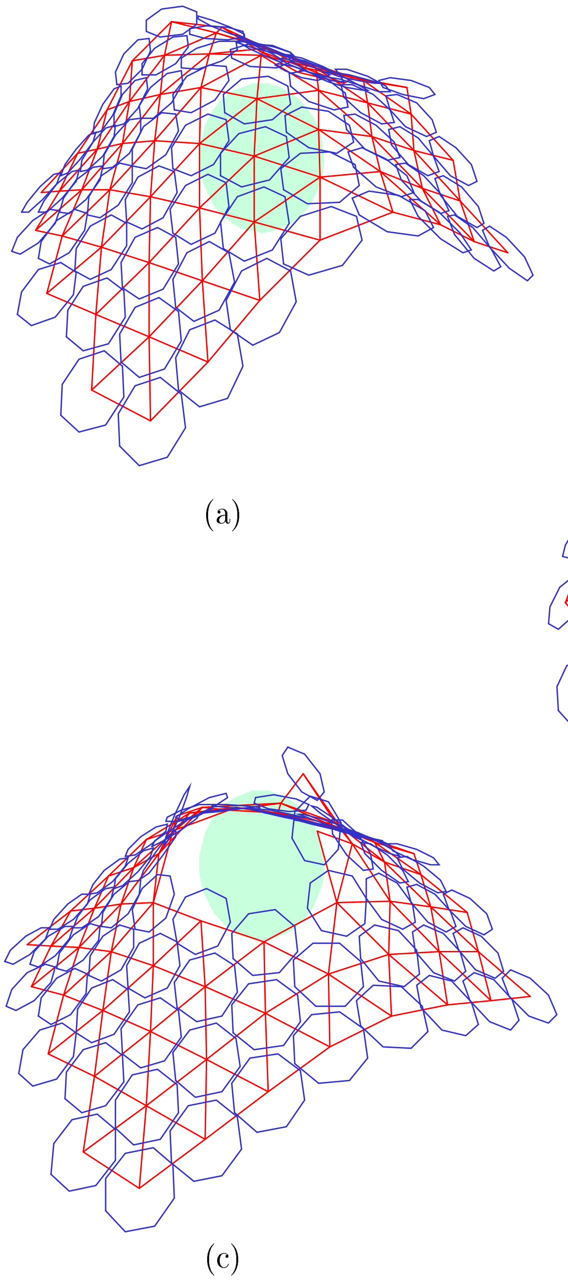


Figure 8.1: Variable physical behavior

Varying the surface characteristics to change the behavior of a sheet of material (a) cloth draping, (b) plastic deformation, (c) tearing.

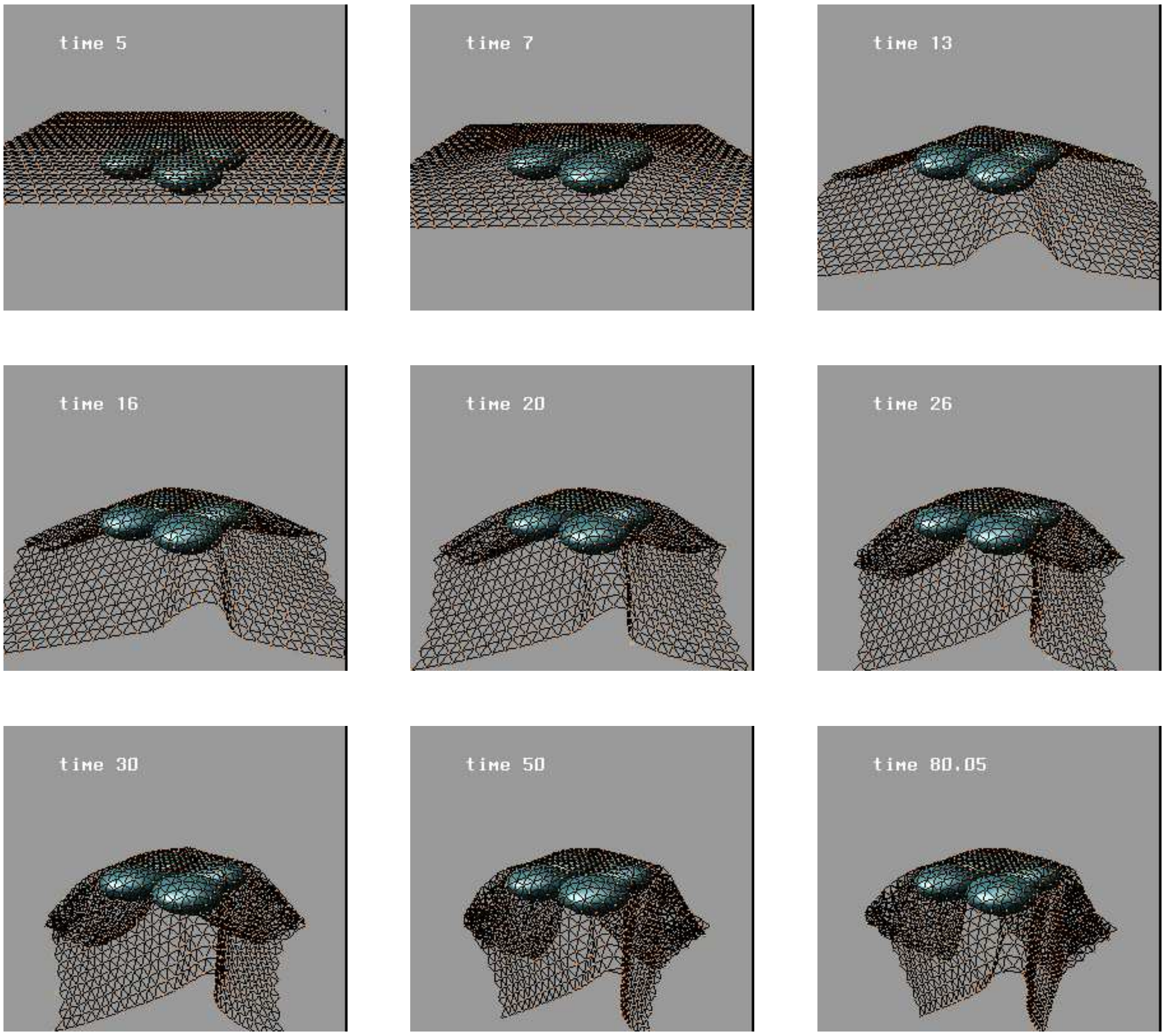


Figure 8.2: Draping cloth

A 32x32 sheet of particles draping over four ellipsoids. The particles nearest neighbor interactions were computed once at the beginning and then the same neighbors we kept throughout the simulation. Thus, in this example, there is fixed coupling much like a spring mass system.

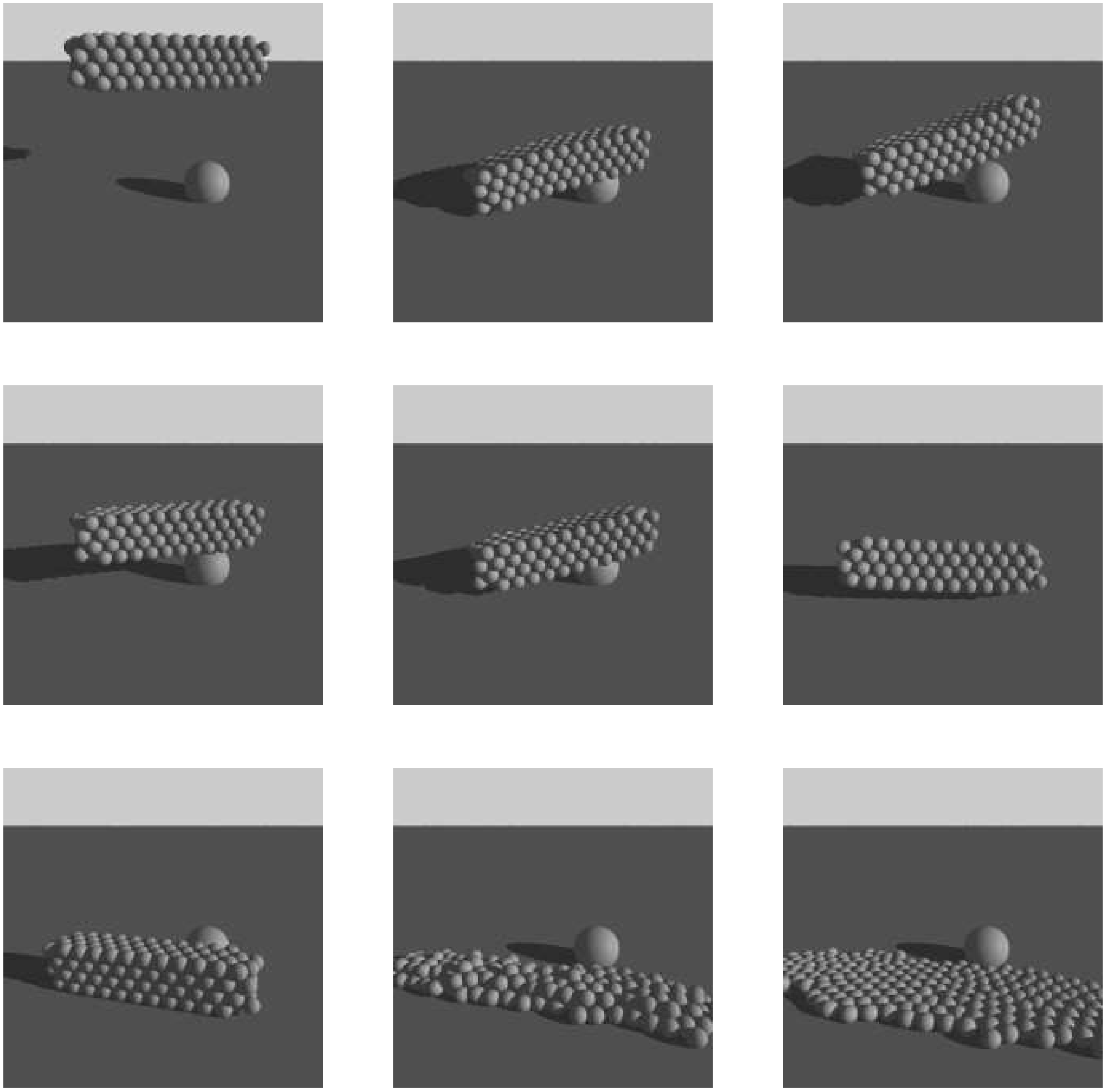


Figure 8.3: Beam colliding and melting

A solid beam falling, colliding with obstacles and later melting. The frames were taken from an animation at the following. times: $t = 0, 3, 4, 5, 9, 12, 25, 70, 83$. The beam falls from a height due to gravity and collides with a spherical object and the ground plane. It deforms slightly and bounces up as seen in the fourth frame. In the fifth frame it collides again and rolls forward (six and seventh frames). In the last two frames it is in the process of "melting" after absorbing heat from the hot ground plane.

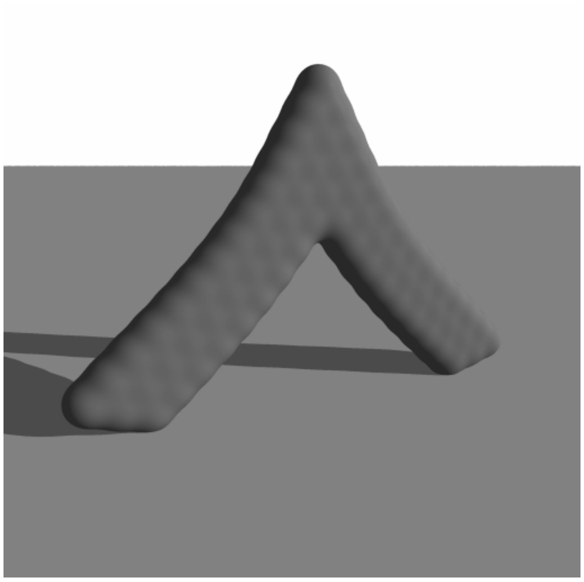


Figure 8.4: Flexible solid

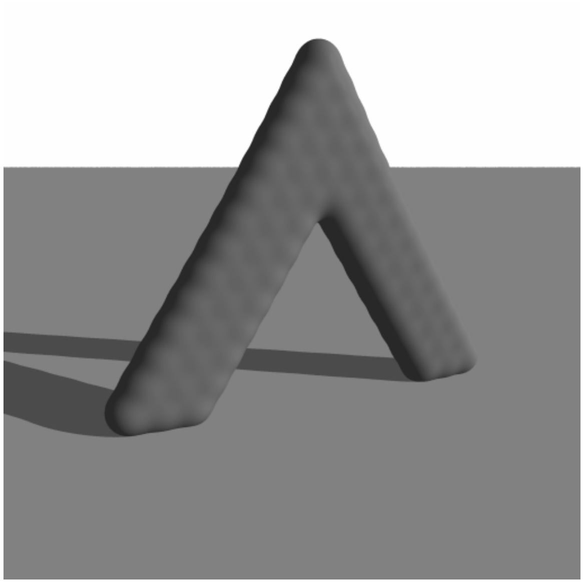


Figure 8.5: Rigid solid

In Figure 8.4 the exponents n and m were set to the values $(n = 4, m = 2)$ and in Figure 8.5 set to the values $(n = 8, m = 6)$.

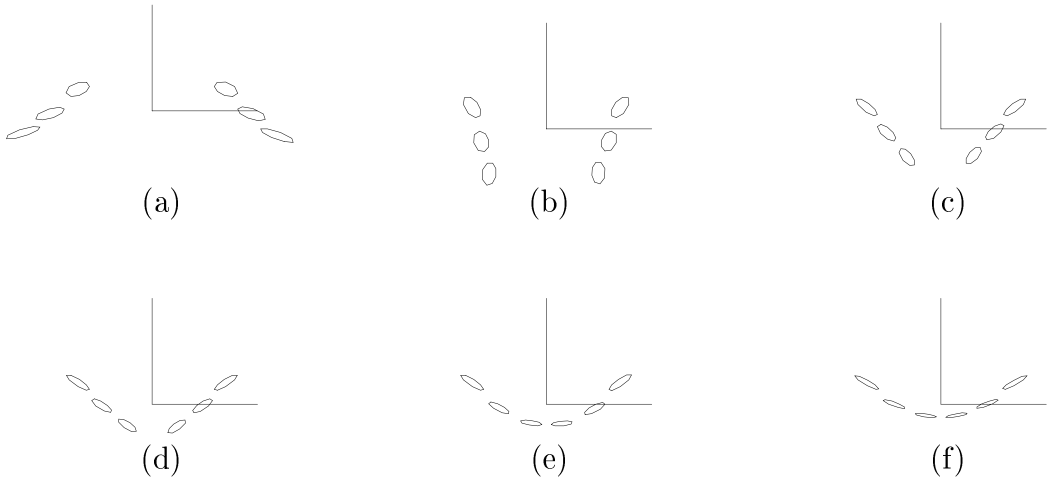


Figure 8.6: Self joining chain under gravity

"cast" solids by pouring liquid particles into a mold and then cool the particles into a solid. A user can "sculpt" a model by interactively adding, deleting, and deforming the model. Local sections of the model can be heated to soften the material similar to how an artist creates a sculpture in wax before casting it in metal.

8.2.1 Basic Modeling Operations

This section describes some basic operations for interactively creating, editing, and shaping particle-based surfaces or solids. The most basic operations are adding, moving, and deleting single particles. One can form a simple surface patch by creating a number of oriented particles in a plane and allowing the system dynamics to adjust the particles into a smooth surface. One 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 2D locator (mouse) to perform 3D locating and manipulation, inferring the missing depth coordinate when necessary from the depths of nearby particles. Adding 3D input devices for direct 3D manipulation (Sachs, Roberts and Stoops, 1991) would be of obvious benefit.

To control the shape more accurately, we can fix the positions and/or orientations of individual particles. Figure 8.6 shows an example of two particle "chains" whose endpoints have been fixed in space.[1] When simulated gravity is turned on, the two chains fall together and join at the bottom due to inter-particle attraction forces. The two chain pieces swing under gravity, and when their endpoints are near each other, they link into a single chain, like a trapeze.

[1]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 *snakes* (Kass, Witkin and Terzopoulos, 1987).

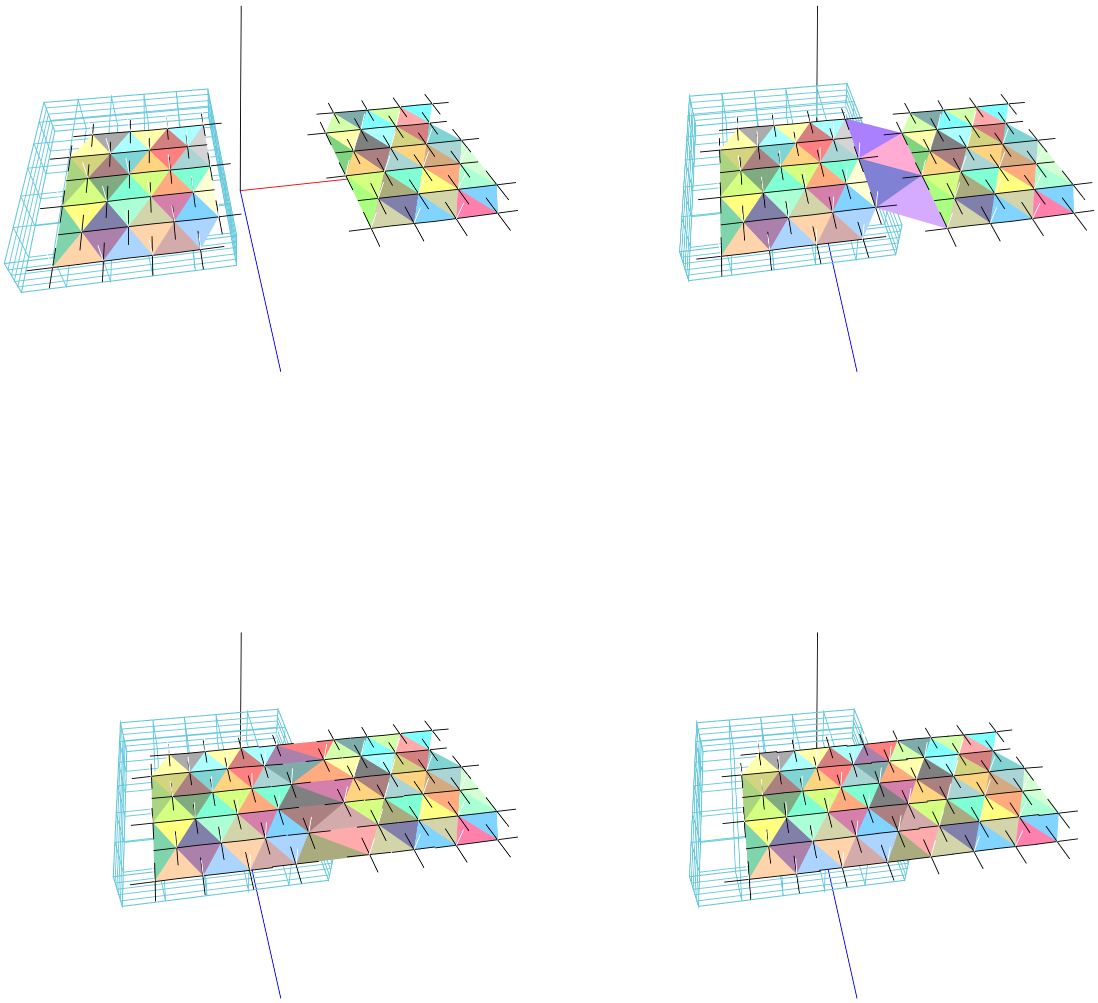


Figure 8.7: Welding two surfaces together.

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

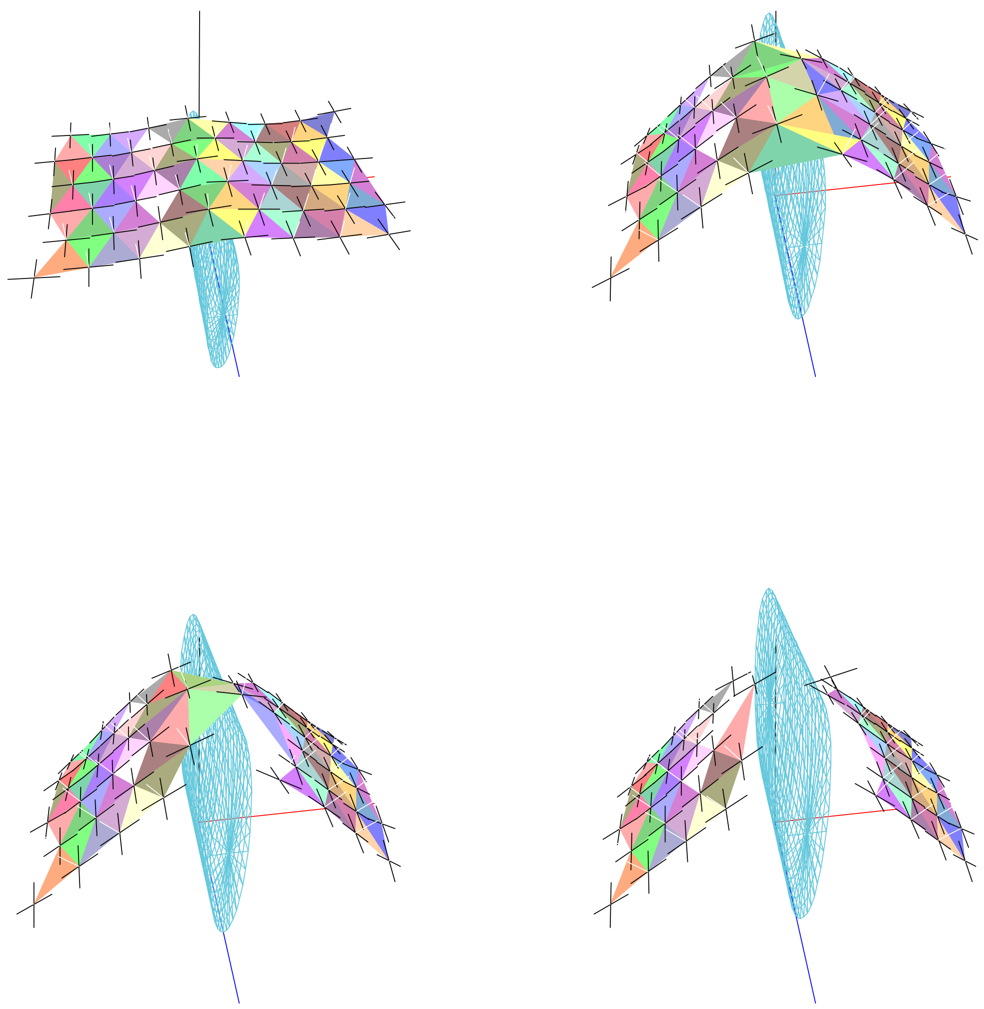


Figure 8.8: Cutting a surface into two.

The movement of the knife edge pushes the particles in the two surfaces apart. The positions of the particles on the left and right edges of the sheet are fixed.

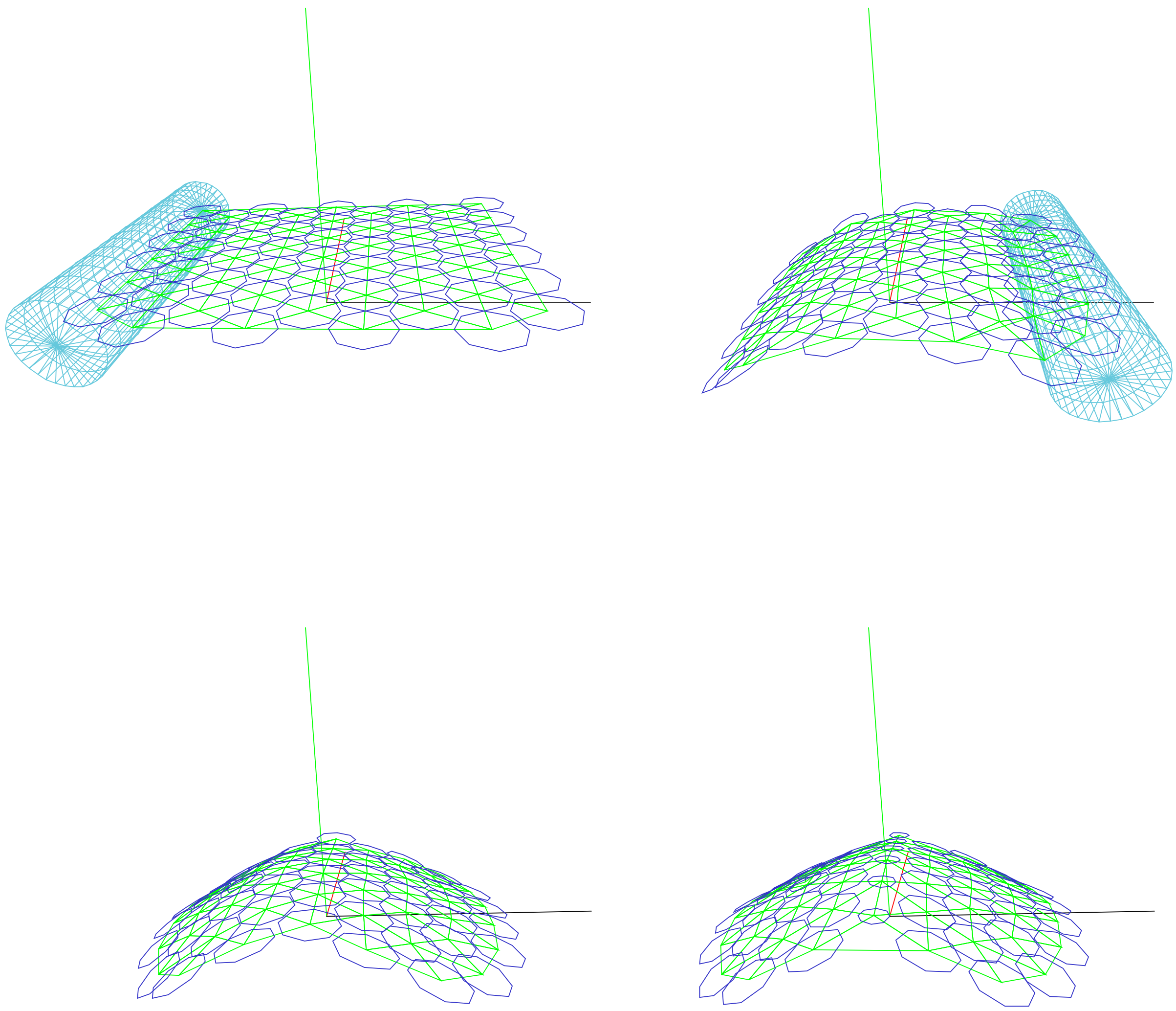


Figure 8.9: Putting a crease into a surface.

The cylinder acts as a moving tool. We grab particles on the left and move them down. these particles are then fixed at this location. Likewise particles on the right side of the sheet are moved and fixed in position. After creating a bent surface we use a "creasing" tool to convert oriented particles into unoriented particles which ignore smoothness forces.

In addition to particle-based surfaces, the 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, as attracting surfaces, as large erasers and as volume-cursors (Zhai, Buxton and Milgram, 1994), which grab all of the particles inside them, allowing groups to be moved at once. Our 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 8.7). Inter-particle forces pull the surfaces together and re-adjust 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 8.8). Here, we use the "heat" of the cutting tool to weaken the inter-particle bonds. 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 8.9). 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).

We have designed a heuristic to control the automatic addition of particles. The rule is 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. This is a reasonable assumption as the dynamics of our energy minimizing system are continually updated during modeling. The (*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 $d_{\min}/2$. 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 8.10, where a ball pushing against a sheet stretches it to the point where new particles are added.

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 when 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.

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 8.11). Forming such a 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

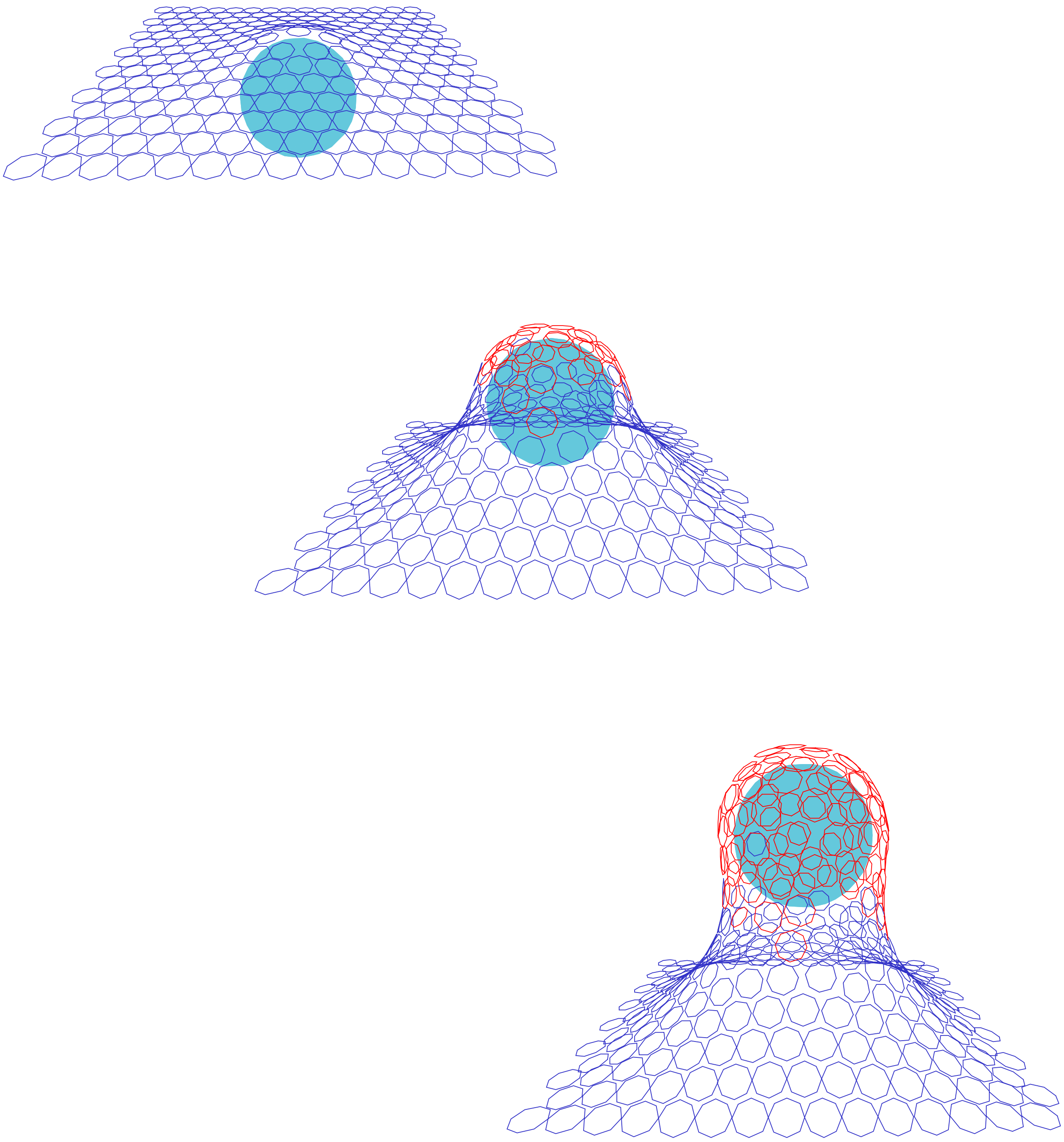


Figure 8.10: Particle creation during stretching

As the ball pushes up through the sheet, new particles are created in the gaps between pairs of particles.

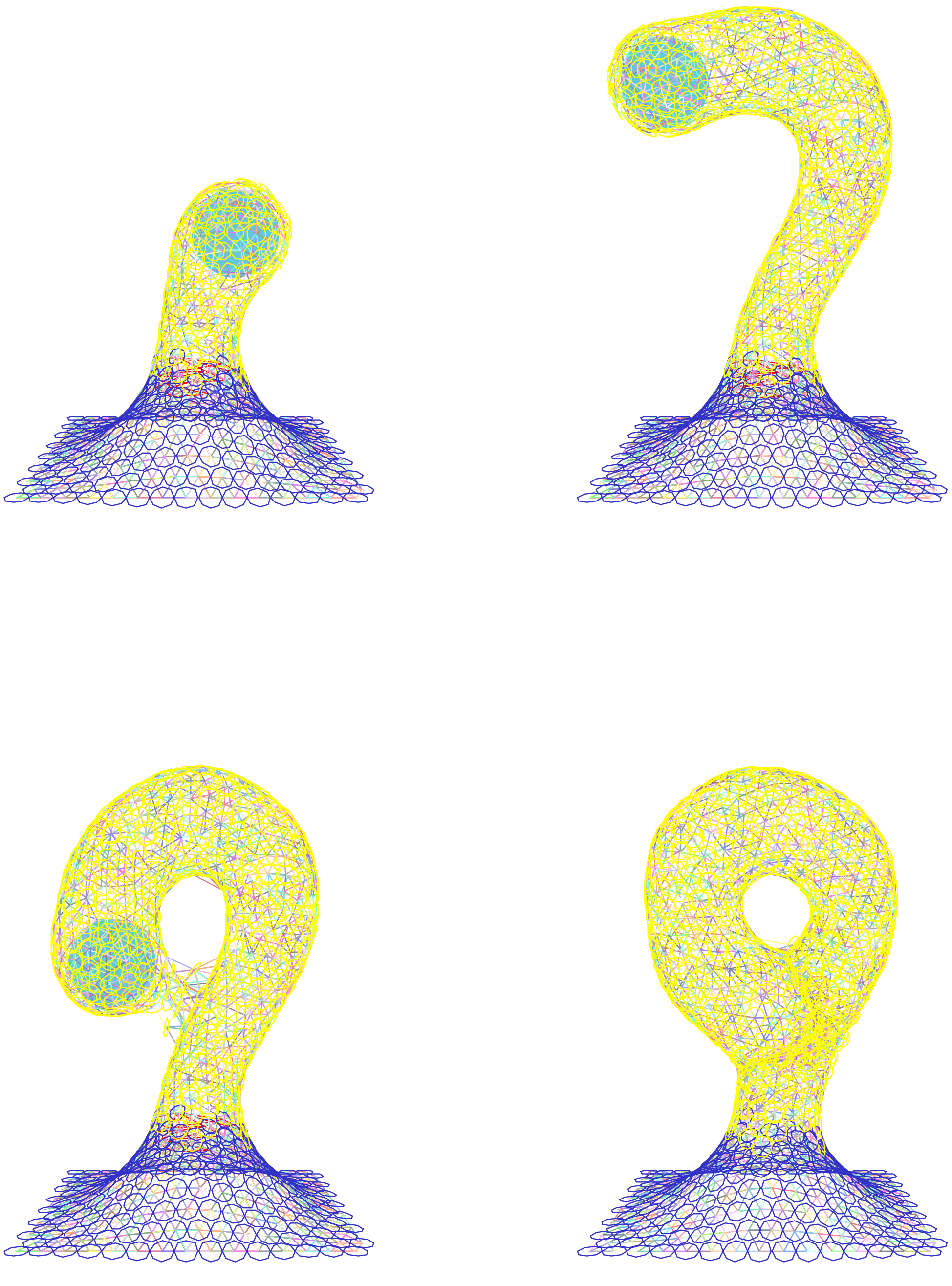


Figure 8.11: Forming a complex object

A complex shape is formed as the initial surface is deformed upwards and then looped around. The new topology (a handle) is created automatically.

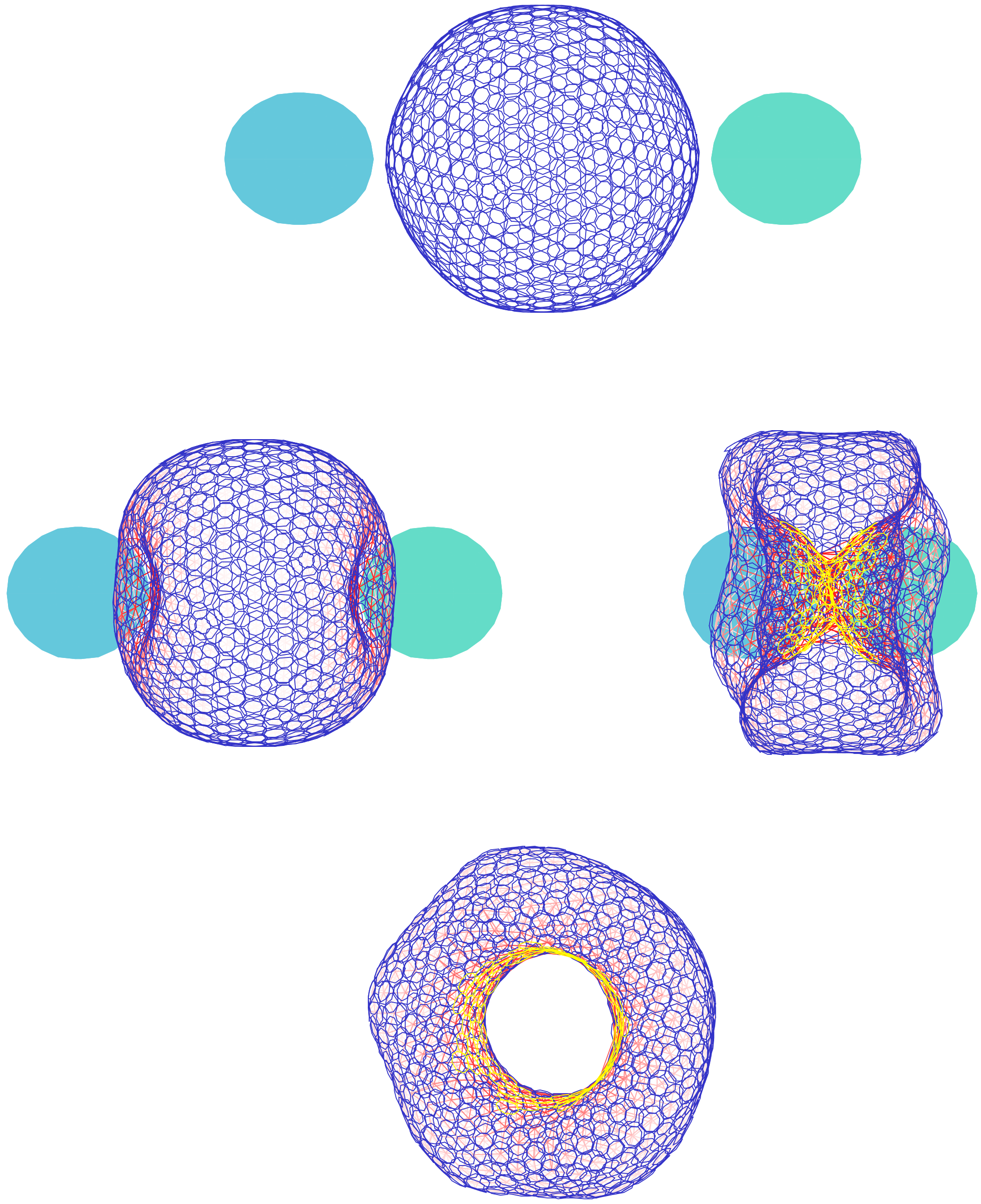


Figure 8.12: Deformation of sphere to torus

We deform a sphere into torus using two spherical shaping tools. the final view is from the side, showing the toroidal shape.

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 (Figure 8.12). New particles are created inside the torus due to stretching during the formation process, and some old sphere particles are deleted when they are trapped between the two shaping tools.

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. 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 the models 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.

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 filleting at part junctions.

8.3 Surface Reconstruction

An important application of our oriented particle systems is the interpolation and extrapolation of sparse 3D data. This is a particularly difficult problem when the topology of the surface to be fitted is unknown. Oriented particles can provide a solution to the unknown topology problem by extending the surface out from known data points. This technique is particularly useful for interpolating sparse position measurements available from stereo or tactile sensing.

The basic components of our particle-based surface extension algorithm are two heuristic rules that control the addition of new particles. The stretching and growing 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 rule, the stretching rule (section 5.2.3), allows particles to be added in openings between two existing particles. 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.

8.3.1 Surface Fitting

With these two rules, we can automatically build a surface from a collection of sparsely sampled 3D point data. 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 completing a rough surface approximation, 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. The fitted surface does not assume a particular topology as many previous 3D surface fitting models have (Terzopoulos, Witkin and Kass, 1988; Miller et al., 1991; Vasilescu and Terzopoulos, 1992).

In Figure 8.13, a toroidal surface is interpolated through a set of seed points. The resulting particle surface can be triangulated to generate a continuous surface description. The toroid was sparsely sampled using only 300 points which is significantly less dense than the sampling in other commonly sampled data (such as range images, height fields, and cat scan volume data sets). Oriented particles have the additional benefit in that they are not restricted by surface topology.

We can also fit surfaces to data sampled from open surfaces, such as stereo range data (Fua and Sander, 1992). 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 8.14 and Figure 8.15). 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 8.14, 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

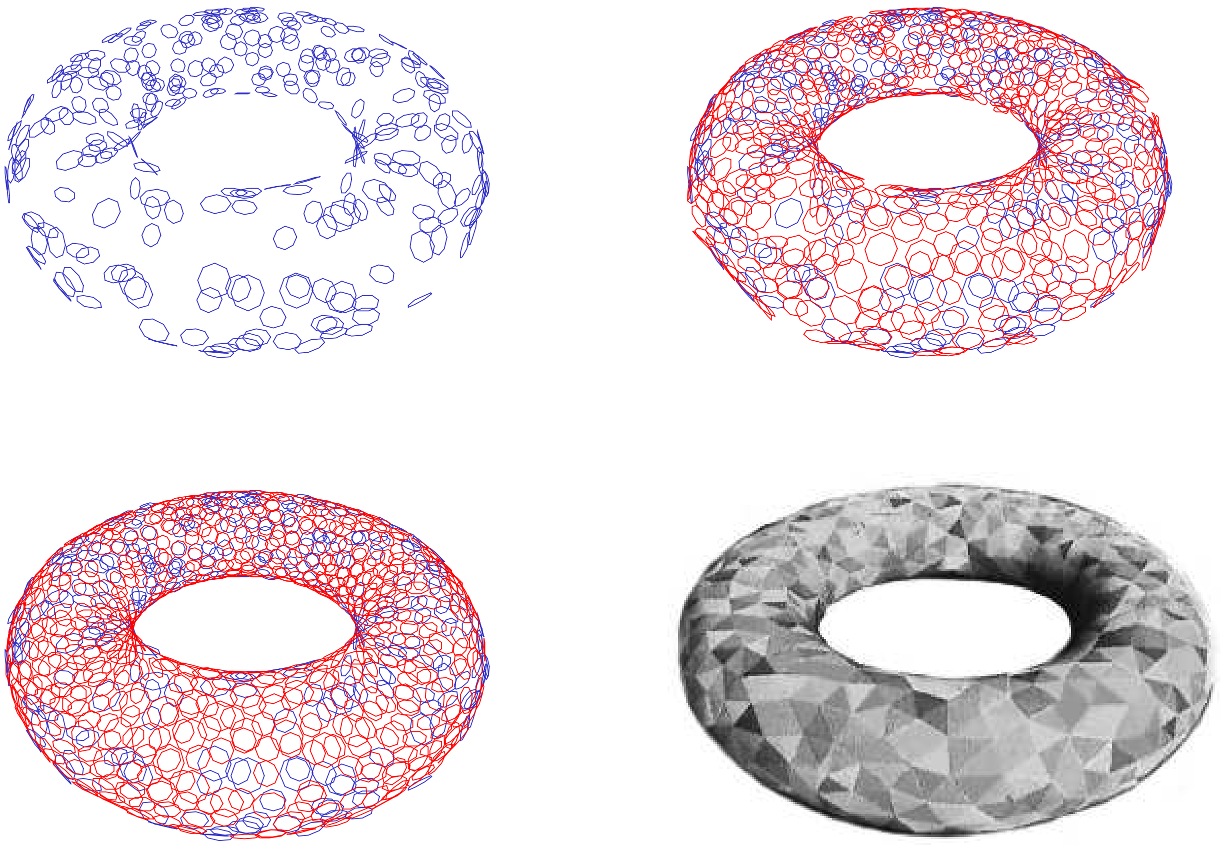


Figure 8.13: Surface interpolation through 3D points

Surface interpolation through a collection of 3D points. The surface extends outward from the seed points until it fills in the gaps and forms a complete surface. When viewed in color, the original points are blue, the interpolated points are red, and the final torus is composed of triangles of various colors.

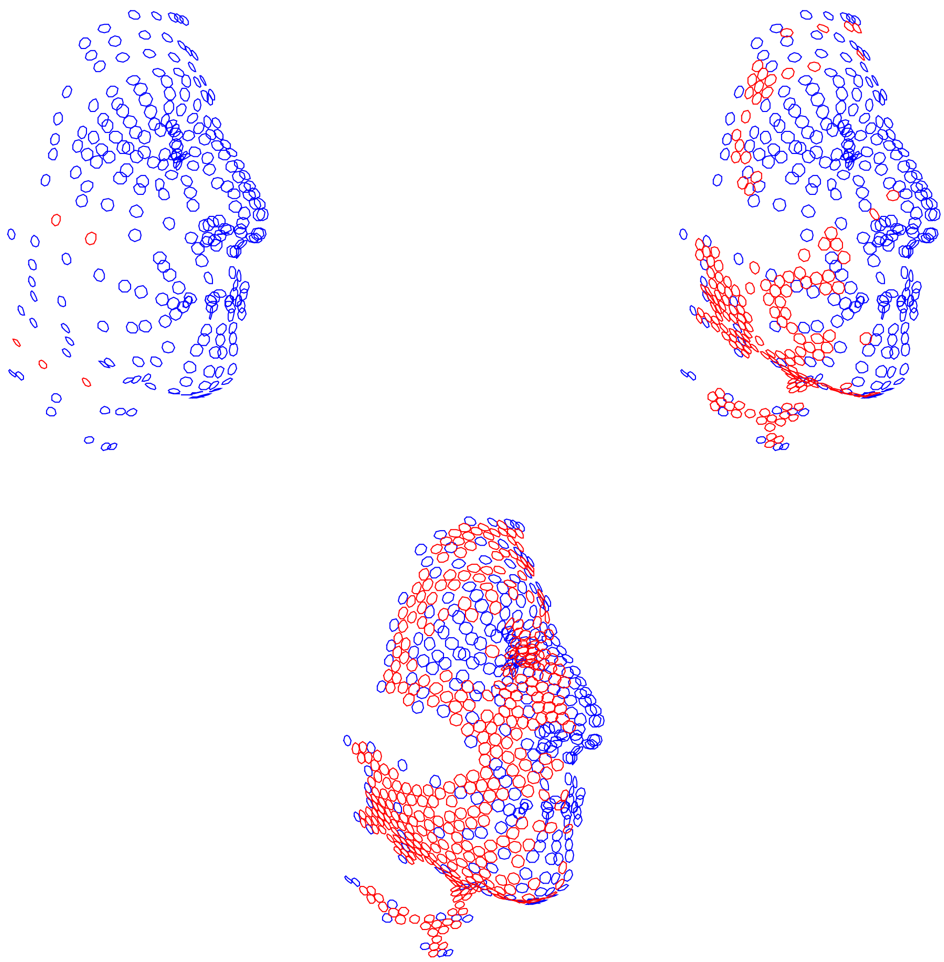


Figure 8.14: Interpolation through 3D points

Interpolation of an open surface through a collection of 3D points. Particles are added between control points until all gaps less than a specified size are filled in. When viewed in color the blue circles are the original data points, and the red circles are the interpolated data points. Increasing the range would allow the sparse areas of the cheek and neck to be filled in, as shown in Figure 8.15.

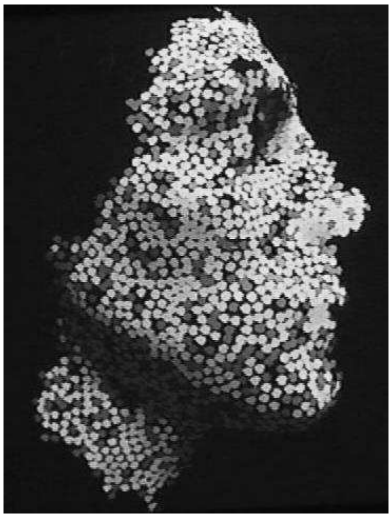


Figure 8.15: Interpolation with increased range

Interpolation of an open surface through a collection of 3D points. We used the same initial data as in Figure 8.14. The difference in results is due to increasing the search range. The dark discs represent the initial data points and light discs represent the interpolated points.

range.

8.3.2 Principal Frames

We can extend our reconstruction process to compute the principal frames and lines of curvature over the surface. A principal frame is defined as a local frame aligned with the principal directions, $\mathbf{e}_1$ and $\mathbf{e}_2$, at a given point (appendix A). The principal directions are in the directions of minimum and maximum curvatures κ_1 and κ_2. We can rotate each particle frame such that local x and y axes align with the directions of principal curvature at that point. These directions can be used to compute lines of curvature, which are useful in computer vision applications. From the principal frames, we can then compute estimates of the principal curvatures.

We can rotate a particle's local frame into a principal frame using a potential function that induces a torque about the local z axis. We can define such potentials using the notation in local coordinates of particle i, as given in section 4.3

$$\begin{aligned}
\mathbf{x}'_i &= [0,0,0]^t \\
\mathbf{n}'_i &= [0,0,1]^t \\
\mathbf{x}'_j &= [x_j, y_j, z_j]^t \\
\mathbf{r}'_{ij} &= \mathbf{x}'_j - \mathbf{x}'_i = [x_j, y_j, z_j]^t
\end{aligned}$$

for example, the potential term

$$\phi_{\mathrm{S}1} = x_j^2 z_j \tag{8.1}$$

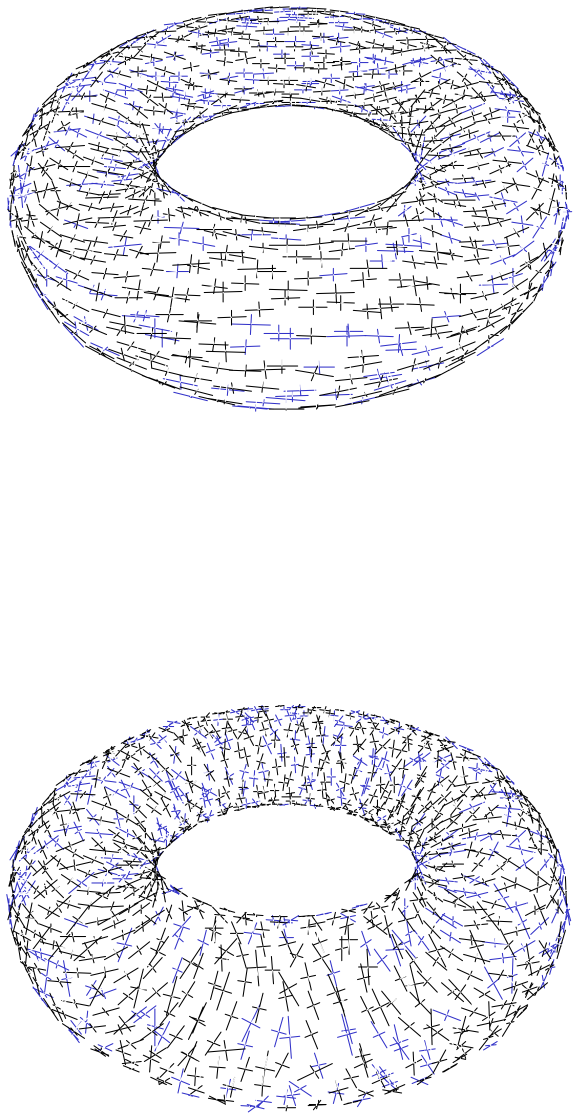


Figure 8.16: Lines of curvature

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 upper and lower images are before and after the addition of ϕ_S.

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

$$\phi_{S2} = x_j^2 z_j^2 \tag{8.2}$$

encourages the x axis to align itself in the direction of the minimum absolute value of normal curvature, i.e. $|\kappa|$. The potentials can also be written in global coordinates

$$\begin{aligned}\phi_{S1} &= (\mathbf{e}_1 \cdot \mathbf{r}_{ij})^2 (\mathbf{n}_i \cdot \mathbf{r}_{ij}) & (8.3)\\ \phi_{S2} &= (\mathbf{e}_1 \cdot \mathbf{r}_{ij})^2 (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2, & (8.4)\end{aligned}$$

where $\mathbf{e}_1 = \mathbf{r}_i[1, 0, 0]$ is the direction of particle i's x axis. 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. Orienting each particle in the system results in a covering of the surface with principal frames indicating the principal directions over the surface (Figure 8.16). The top Figure shows particles with random twists about the normal. The middle Figure shows particles aligned as principal frames. The bottom Figure is drawn such that the length of the axes correspond to the magnitude of curvature computed.

8.3.3 3D Volume Segmentation

Our surface fitting algorithm may be used to help segment structures in 3D volumetric data such as CT, MRI, or other 3D medical imagery. To perform this segmentation, we first apply a 3D edge operator (Monga et al., 1990) to the data and use the edges to initialize and attract particles, or directly use gradients in the 3D image as external forces on the particles. In this application, our 3D surface model can be viewed as a generalization of the active deformable surface model (Terzopoulos, Witkin and Kass, 1988; McInerney and Terzopoulos, 1993), but without the restrictions imposed by a manually selected surface topology.

Figure 8.17(a)–(c) shows slices from a CT scan of a plastic "phantom" vertebra model (decimated to $120 \times 128 \times 52$ resolution). Figure 8.17(d) shows the reconstruction near completion. Figure 8.17(e) shows a Gouraud shaded rendering of the reconstructed surface. This smooth, triangulated model contains 6,650 particles and 13,829 triangles, and was created by seeding a single particle and extending the surface along high 3D edge values until a closed surface was obtained.

8.3.4 Surfaces From Silhouettes

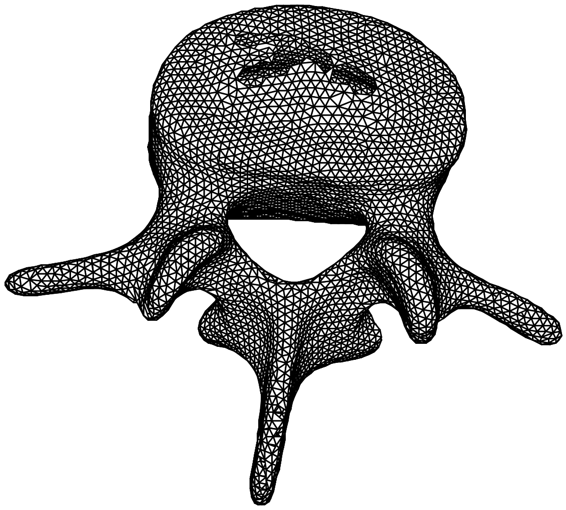
We have applied our particle-based approach to the reconstruction of triangulated surface models from the output of a shape-from-silhouettes algorithm (Szeliski, 1991). The algorithm constructs a bounding volume for the object by intersecting silhouettes from a sequence of views taken around an object—in this case, a cup—as it rotates on a turntable. The algorithm represents the volume using an octree (Samet, 1989) (Figure 8.18a).

(a)

(b)

(c)

(d)



(e)

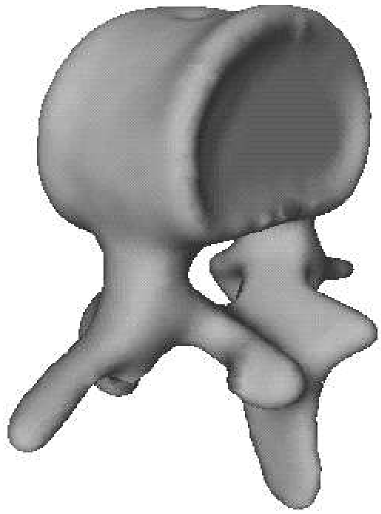


Figure 8.17: Reconstruction from CT volume data

3D reconstruction of a vertebra from $120 \times 128 \times 52$ CT volume data: (a) xy slice, (b) xz slice, (c) yz slice, (d) reconstructing a 3D surface model with triangulated particles, (e) shaded surface.

To reconstruct the surface model of the cup, we first create a volume occupancy array from the octree representation and then apply the 3D edge operator and our reconstruction algorithm as in the vertebra example. Figure 8.18b shows the reconstructed model of the cup. The reconstructed surface has 3,722 particles and 7,568 triangles.

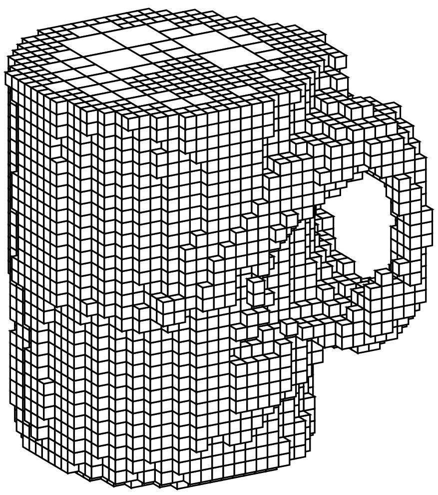
8.4 Summary

We have applied our dynamically coupled particle system to three different problems, that of computer assisted animation, free form surface modeling, and surface reconstruction. We briefly summarize.

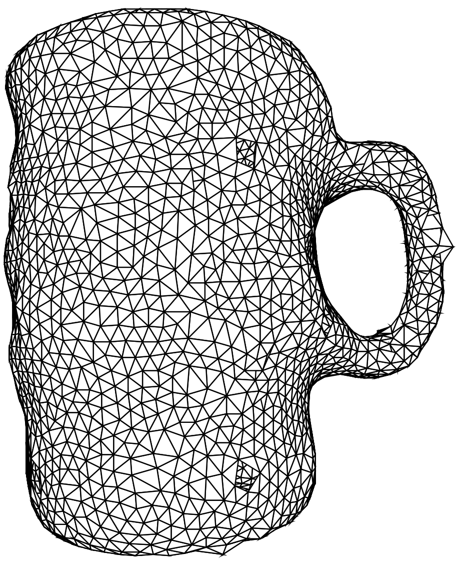
For animation, our model provides the ability to model from rigid to fluid like behavior. By changing the parameters of the Lennard-Jones potential we can change how rigid and how brittle our objects are. We have included a model of heat into our system allowing objects to be frozen and melted. Our model exhibits behavior similar to physically based finite element approximations for small deformations. For large deformations, we simulate tearing and infinitely stretchable materials. For rigid body dynamics and modeling small deformations, our model is computationally more expensive than specialized techniques limited to these behaviors. In contrast, our model exhibits a wide range of physically inspired behavior, allowing gross changes in topology.

For free form surface modeling we have implemented an interactive system for creating, editing, and shaping surfaces. We can add, delete, and move single particles. In addition, arbitrary polygonal models can be used as tools to select and move groups of particles, to act as large erasers, and to repel or attract particles. Particle surfaces can be "cold welded" together or cut into separate pieces. Groups of particles can be constrained in position and orientation similar to defining character lines of variational surfaces. Local curvature discontinuities in the surface can be encouraged by "unorienting" lines of particles, such that the discrete curvature energies are not applied to those particles. Heat can be applied to create more malleable areas, enhancing local shaping operations. The disadvantages of our model is it does not enforce analytical constraints such as strict smoothness criteria found in spline models, and it is more expensive in terms of data and computation time than strictly geometric techniques. Our model does possess distinct advantages. Particle based models are easy to shape, extend, join, and separate. Adjusting the relative strengths of the potential functions varies the surface's resistance to stretching, bending, and variation in curvature. Large changes in surface topology and genus can be easily achieved with minimal user interaction, allowing complex natural looking shapes to be easily constructed. Our surfaces incorporate physical properties modeled over time and thus react to user manipulation in a natural and intuitive manner.

For surface reconstruction our particle model can reconstruct surfaces of arbitrary genus from a variety of 3D data. We can interpolate surfaces through sets of sparse 3D data points to reconstruct both open and closed surfaces. By extending our surface from known data points, we can reconstruct closed surfaces of arbitrary topology.



(a)



(b)

Figure 8.18: Reconstruction from silhouettes

Reconstruction of a surface model of a cup from silhouettes: (a) cup bounding volume represented as an octree, (b) triangulated surface of reconstructed model.

To reconstruct open surfaces from sparse 3D point sets, we interpolate the original surface by inserting particles between known sample points. After a rough surface shape is achieved, we can release the original sampled particles to smooth the final surface, thereby attenuating the original samples. Our surface fitting algorithm can be used to help segment 3D volumetric data such as CT and MRI data. By attracting particles to an edge filtered version of the volume data or by following gradients in the original image, we can construct surfaces which segment the data into sets. This can be viewed as a generalization of iso-surface polygonization. Our technique could easily be applied to generating surface descriptions of iso-surface fields. Our approach can be applied to other types of 3D data, such as volume occupancy arrays generated by shape-from-silhouette algorithms. The disadvantages of our model are that it is not guaranteed to reconstruct the correct surface if the original data sampling is highly anisotropic, and that it may be more computationally expensive than using algorithms designed for a known surface genus and topology. The advantages of our model is that it can reconstruct continuous connected surfaces of arbitrary topology and genus even when this information is unknown. It is an optimal surface fitting procedure minimizing selected energy functionals, and the degree of smoothness can be controlled by weightings of the potential functions. In addition, principle frames can be computed over the final surface, which can then be used to compute lines of curvature over the surface and estimates of the principal curvatures.