Dynamically Coupled Particle Systems for Geometric Modeling, Reconstruction, and Animation

by

David Love Tonnesen

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

This dissertation presents a new technique, based on dynamically coupled particle systems, for creating and manipulating complex three dimensional shapes in a fluid like manner. The most novel feature of this approach to shape representation is the use of self organizing primitive elements. In the simplest case, these primitive elements, or particles, each posses state variables of position and mass, and the system of elements interact through pairwise potential energy functions. More complex systems include additional state variables combined with simple heuristics to create application specific behavior. The ability of these systems to self organize provides a representation technique which exhibits dynamically changing structure, an attribute not found in popular spline and polygonally based representations. To illustrate the usefulness of this approach it is applied to the following problems: free form shape modeling, computer assisted animation, and surface reconstruction. For free-form modeling the approach supports smoothness constraints similar to those inherent in the deformation energies of popular, elastic surface models. Unlike spline patches or parameterized surface models, the model does not attempt to enforce analytical continuity conditions, such as tangent or curvature continuity over the surface. Applied to computer assisted animation the approach computes the movement and deformation of models mimicking, at a rudimentary level, the physical behavior of flexible solids and fluids. Applied to surface reconstruction, these systems can infer surface structure from sparse data sets, without a prior knowledge of the surface structure or the topological genus. In summary, dynamically coupled particle systems provide an useful alternative to traditional shape representation and manipulation techniques.

Acknowledgments

I thank my advisors, Demetri Terzopoulos and Richard Szeliski, for their continued support and advice. I thank Bill Buxton, Eugene Fiume, Michiel van de Panne, and James Stewart, for participating as members of my Ph. D. committee and for their insights and advice. I thank Gavin Miller for fulfilling the role of external appraiser. I also thank the many members of the Dynamic Graphics Project for their support, friendship, and inspiration.

I thank the University of Toronto, Department of Computer Science and the Digital Equipment Corporation, Cambridge Research Lab for the financial support and the facilities used to pursue this research.

I thank my family and my friends who have given me moral support, counsel, and friendship. I thank Bob, Diane, Fabio, Felix, Goofina, Huleo, Jason, Jay, Kathleen, Litsa, Rakesh, Rania, Ray, Rebecca, the Sanchez family, Spider Dave, Steve, Tasso, and William. And in particular, I thank Jos, Melanie, Pam, Rakesh, Rod, and Sylvia.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

There is an endless variety of shapes in the world; from the simple Euclidean shapes such as rectangles, spheres, and cylinders, through the smooth swept curves of a boat hull, to the fractal nature of clouds, mountains, and trees. If the computer is to assist us in our endeavors to represent the world, it is important that we can represent and manipulate the wide range of shape.

The requirements for a suitable shape representation and for tools to manipulate such representations vary according to application. Industrial designers require exact analytical control of shape. In automotive, aerospace, and marine vehicle design, sculptors require control of the "curvature" and "fairness" of a surface. Character animators in the entertainment industry require "expressive" control of their creations. The special effects industry requires emulations of natural effects to create rich and complex visual elements. Robotics require spatial shape representations that can be derived from the environment and then used to navigate through the environment. Medical scientists require surfaces that segment data collected from non-invasive sensing techniques, such as CAT and MRI scans. The physical sciences require surfaces that provide an "optimal fit" to empirical data. Traditional computer based tools for representing and manipulating shapes both limit and enhance the ability to create specific classes of shapes.

In computer graphics, a common trait among shape representations is that the global structure is explicitly defined. In the case of the classical primitives (spheres, cubes, superquadrics) the structure is defined by the underlying parameterized equations. In the case of splines and polygons, structure is defined by manually connecting surface patches. And in the case of skeletal implicit surfaces, the structure is defined by the underlying skeleton structures. While these shaping representations are excellent at creating a wide variety of shapes, large changes in the surface structure and changes that do not preserve genus require significant user interaction.

In this dissertation, we investigate the problem of developing a shape representation in which the structure of a shape is inferred, rather than explicitly specified. By removing the restrictions on structure, we open the door to new ways of both creating and manipulating shapes. Our investigation takes us away from a purely geometric approach to modeling to one that combines geometry and physics.

Our goal is to develop a new shape representation model, based not on an underlying parametric representation, but rather on systems of self-organizing elements from which shape and structure emerge. We illustrate our approach by applying it to three common tasks: free-form shape modeling, computer assisted animation, and surface reconstruction. To this end we have three auxiliary goals:

- To create and manipulate complex shapes, not by the traditional method of manually defining the arrangement of surface patches and continuity conditions, but rather through an intuitive and flexible sculpting metaphor where "physical" tools are used to manipulate synthetic materials.

- To show that these shapes can be animated over time, not by key framing, but by mimicking the physical behavior of various materials.

- To reconstruct and extract surfaces of arbitrary topology from three dimensional data.

1.2 Application Areas

Our three auxiliary goals apply to the following areas.

1.2.1 Shape Modeling

There are two basic approaches to the modeling of shapes. The first is to create a shape to meet a known set of specifications. Another approach, more exploratory in nature, is to evolve a shape from simple to complex, until it is aesthetically pleasing. In this case the focus is not on measurable analytic qualities but on subjective aesthetic qualities. Spline based interpolation, extrusion, and constructive solid geometry, enforce analytical constraints on a shape, aiding in the objective design approach. The success of analytically based tools are in part because the constraints of each technique assist in defining a particular class of surfaces. However, if used to create shapes outside of the intended domain, the same constraints may hinder the creation process by imposing unnecessary limitations. For the exploratory approach, the design tools must be flexible in the sense that the constraints on the *design process* should be minimized. Since numerous computer based tools exist to assist in the creation of shapes with strict geometrical properties, good for precise analytical control of a surface, let us instead consider what properties are useful to designers who follow a more exploratory approach.

The occupation of the sculptor, whether working in wood, clay, plaster, wax, glass, or stone, has existed for centuries. The medium that these artists have chosen to realize their creations have been chosen for their long term durability, and for the material characteristics which allow the artist to shape and reshape the material to its final form. Perhaps some of the needs of the exploratory design process can be observed from considering the materials and tools sculptors have traditionally chosen.

Sculptors often design their shapes using a malleable material, such as wax, which can be easily sculpted with a small collection of tools. The shape can be built up or carved away, thereby providing flexibility to the creation process. After the design is complete, the sculpture is realized in the more expensive and less flexible materials of cast metal and stone. Sculptors use glass to create smoothly curving surfaces. By heating the glass, the artist changes the physical properties so that it can be shaped without breaking. A common technique is to stretch the glass, either by directly pulling the glass or using the force of breath to “blow” the glass into a smooth flowing shape. In industrial applications, smoothly curved lines and surfaces are often designed by bending strips of metal, exploiting the inherent nature of the material. The ceramic sculptor has perhaps the widest range of sculpting methods, which is due to the nature of clay. Beginning with a lump of clay they can stretch the clay, deform it with pressure, break or cut pieces of clay off of the main body, join pieces of clay together adding handles and loops, and build up features by incrementally adding clay to an existing form. If we can incorporate some of these qualities into our shape design algorithms we will provide the user with powerful metaphors to use in sculpting shapes on the computer.

This premise is supported by the fact that several current computer techniques mimic techniques used by sculptors of physical materials. In computer graphics, spline curves are used to enforce the continuity conditions of metal splines traditionally used in the ship building industry. Surfaces of revolution are used to create curved surfaces symmetric about an axis, like shapes thrown by a potter at the wheel or created by a carpenter at a lathe. The ability to add and subtract geometric objects from one another are implemented by constructive solid geometry (CSG) and implicit surface formulations. Still, many of the techniques used by the sculptors of real world materials are unavailable for the computer based sculptor; and when they do exist, they usually do not occur within the same shape representation.

For a metaphor to be effective we do not need a formal mapping from the metaphor source to the target (Madsen, 1994). For example, the piecewise spline model is based on the behavior of metal splines, yet the computer spline does not need to be as cumbersome to manipulate and store as long metal strips. To effectively use the metaphor of sculpting with clay we do not necessarily want to incorporate the fact that clay must be kept moist while it is being formed to prevent the clay from cracking and crumbling. Likewise in the molten glass metaphor we do not have to simulate the fragile nature of cooled glass. In creating a metaphor, undesirable qualities should be ignored, while desirable qualities enhanced.

For shape design and rapid prototyping applications, we propose an interactive system, based on a sculpting metaphor, which alleviates the designer's need to think about the underlying representation or be limited by its choice (Sachs, Roberts and Stoops, 1991), similar to how a child can sculpt clay while ignorant of the clay's chemical structure. Borrowing from traditional sculpting materials, we propose creating synthetic shaping materials that can be manipulated in a variety of ways. The materials should provide the user the ability to cut, merge, and join shapes; to deform shapes; to carve away and add new material; and to shape and reshape the material with a basic set of tools.

1.2.2 Computer Assisted Animation

In animation, a story is brought to life as a stream of images and sound. Creating the images requires capturing all of the visual changes that occur in the story, including the shape, color, transparency, structure, texture, time-vary position (motion-dynamics), and lighting of objects (Foley et al., 1990). Not surprisingly, animation is expensive and time consuming. The quantity of visual information that must be created, manipulated, and stored is enormous. While short animations, such as television cartoons and commercials, are produced in only a few months, longer feature length animations require three to five years of effort. It can be argued whether or not the computer has sped up the process of creating animation, but it cannot be argued that it has had no impact. The computer has come to assist the artistic process, by both reducing the time spent per image and expanding the range of images routinely created.

Three dimensional computer animation is the art of modeling shape changing over time. To express intent and reaction, the animator needs ultimate control over the shape. Because of this, specifying the object shape at key frames has been the dominant method used. Interpolating between key frames parallels the job of the inbetweening animator in traditional animation. Inverse kinematics is coming to play a bigger role in animation, because instead of simply interpolating, it computes the movement of jointed figures. Recently, physics based simulations have been making their way into the animation and special effects businesses. In these simulations, control is traded for dynamic complexity and visual richness, making them attractive alternatives to more time consuming techniques.

For simulating secondary action, the effects animator needs not only a system which aids him in representing shapes, but also the flexibility to change the shapes and their behaviors. We propose a physically inspired approach that allows one to sculpt a variety of shapes, to animate the shapes over time through a physical simulation, and to varying the physical properties associated with the shape.

1.2.3 Surface Reconstruction

Large quantities of data are routinely generated by computer simulations and collected by non invasive sensing devices (Ney, Fishman and Magid, 1990; Stytz, Frieder and Frieder, 1991; Higgins, Chung and Ritman, 1990; Wolfe Jr. and Liu, 1988; Baker, 1988). The sheer quantity of the data collected presents a fundamental problem when it comes to interpretation. The human visual system is considered a high bandwidth channel for receiving information (Haber and McNabb, 1990; Hibbard and Santek, 1989), making visualization a powerful data presentation tool. Often specific sections of the data are of interest and displaying all of the data, as in many volume rendering techniques, distracts the user from the meaningful features. Such features can be extracted by fitting surfaces to them.

Other interesting uses of surface fitting are to recreate surfaces scanned by laser range finders or from video cameras (Szeliski, 1991; Szeliski, 1993). In the future, geometric modeling may be simplified by entering geometric descriptions that consist of "showing" the object to the computer. This would allow users to send "3D faxes" by scanning an object in one place and interactively viewing a full 3D model at a remote location (Carlbom et al., 1992).

Unknown object topology poses difficult challenges to surface reconstruction that have until recently been largely ignored in the vision literature. Unfortunately, vision systems that must derive quantitative models of complex real-world objects from multiple views cannot avoid the issue of unpredictable topological structure. This is also an important concern in the related fields of biomedical and geological imaging where there is a need to analyze three dimensional arrays of volumetric density or reflectivity data. These arrays are like blocks of marble with meaningful embedded structures. For further analysis, the often complex and topologically unpredictable surfaces of these structures must be extracted and represented as compact geometric models (McInerney and Terzopoulos, 1997). To solve these problems we propose to use a system of self organizing primitive shape elements which react to both the content of the data and neighboring primitive elements. In such a system the topology is not defined by a surface parametrization, but by the relative spatial position of the primitive shape elements.

1.3 Challenges

In creating our new shape representation technique, we face several challenges:

Changes in Structure

Traditional shape modeling techniques are primarily based on the user manually defining the structure of the object. Thus, gross deformations and changes in genus require significant user interaction. The challenge is to create a model where large changes in shape can be made with minimal user interaction. For free-form modeling, we want to be able to easily sculpt a shape without having to consider the underlying structure. For computer assisted animation, the shape structure should change in a natural manner as dictated by the simulation. For surface reconstruction, the technique should be able to infer unknown surface structure from 3D data sets.

Synthetic Materials

Our modeling paradigm is based on the metaphor of synthetic materials endowed with physical properties. For free-form modeling, the synthetic materials should have properties similar to materials used in designing sculptures, such as wax, clay, and glass. For computer assisted animation, the synthetic materials should be able to exhibit a wide range of dynamic and elastic properties and interact with their environment. For surface reconstruction, the synthetic materials should be influenced by the data sets and be able to simulate the optimal fitting properties such as those of a membrane or thin plate.

Customization

To obtain a general shape representation, we need to customize the behavior of the synthetic materials. Some desirable properties include: conservation of volume, conservation of surface area, mass, elasticity, stiffness, and fluidity. For different applications we should be able to select and enforce desirable properties.

Efficiency

The computational cost of the model should be tractable. As the complexity of the shape grows, the time and space costs should not grow exponentially.

1.4 Methodology

We address the challenges with the following approaches.

Criteria

The measure of quality we use for our shape representation is not based on the realism and accuracy of the synthetic material to real world materials, but whether the synthetic material exhibits properties and behavior that assist the application. For example, in a free-form modeling application based on the metaphor of sculpting a malleable material with tools, the synthetic material does not need to exhibit all the properties of say, clay or wax, only a select few which support the metaphor.

Structure

We address the issue of defining structure using discrete shape elements that can be arranged and rearranged into new shapes. The arrangement and spatial proximity between elements defines the surface structure and hence the topology and genus. Useful arrangements of the shape elements is achieved by using a self organizing system. The ability of the shaping elements to self organize, allows us to create shapes that are not based on parametric patches or a prescribed structure. The global structure of the shape emerges from the local physics based interactions of elements.

Particle Systems

We base our modeling representation on particle systems and Newtonian dynamics. Each particle is a discrete shape element; either a volume element or a surface element. Interactions between particles are based on energy functions and local geometric properties allowing for the creation of synthetic materials, that interact with their environment. In this dissertation we solve for a second order dynamical system for applications requiring realism such as animation. For other problems, such as solving an optimization criteria, a first order system is adequate and is a simplification of the more complex second order dynamical particle system we present.

Surface Descriptions

Surface descriptions are based on the arrangement of the shaping elements. For shapes described by volume elements, we use an implicit surface formulation. For shapes described by surface elements, we generate a polygonization based on the spatial proximity between elements.

Efficiency

We maintain efficient computation by insuring that the computational costs do not grow excessively for large systems. We convert a general $O(N^2)$ particle system computation into $O(N \log N)$ expected computation for surfaces, and $O(N)$ expected computation for volumes, by reformulating global computations into local computations. Key to this approach is being able to rewrite energy functions in local terms of each shape element, and the use of hierarchical tree structures to partition space into local regions.

Customization

Particle systems are customized to a given application by designing new inter-particle potentials, application specific forces, and specialized particle creation heuristics. For example, for modeling volumes we use a potential energy function which encourages the equal spacing and preservation of volume, and velocity based force functions to model inelastic deformations. For modeling surfaces we introduce curvature based potential energies to encourage particles to arrange into surfaces. By varying the weightings of these potentials we can encourage more or less smoothness of the surface. Particle creation heuristics are designed to allow surface modeling operations which "stretch" the surface, and surface reconstruction applications to interpolate between sparse 3D point samples. Forces are designed to provide interaction with objects external to the particle system, such as colliding with objects in an animation, interacting with shaping tools in free-form modeling, and attracting particles to "edges" in surface reconstruction applications.

1.5 Contributions

We close this chapter by summarizing our contributions.

1.5.1 Dynamically Coupled Particle System

This dissertation presents a dynamically coupled particle system for shape representation and manipulation. This research has been published in the computer graphics literature (Tonnesen, 1991; Szeliski and Tonnesen, 1992) and in the computer vision literature (Szeliski, Tonnesen and Terzopoulos, 1993b; Szeliski, Tonnesen and Terzopoulos, 1993a).

A novel feature of this approach to modeling shape is the use of a particle dynamics simulation in which particles interact through potential energy functions to create volumes and surfaces. The use of a spatially symmetric potential energy encourages particles to arrange into tightly packed orderings, good for modeling volumes. This basic model is then extended to develop oriented particles, a particle system which encourages particles to arrange into surfaces rather than volumes.

In the oriented particle model, each particle represents an oriented trihedral coordinate frame. Based on these frames, we design new interaction potentials which favor locally planar or locally spherical arrangements of particles. Thus, the oriented particles support smoothness constraints similar to those inherent in the deformation energies of popular, elastic surface models. Unlike spline patches or parameterized models, the model *does not* attempt to enforce *analytical* continuity conditions, such as C^1 tangent or C^2 curvature continuity, over the surface. Instead of a parametric approach to shape modeling, our research uses collections of primitive self-organizing elements from which geometric structure evolves as the system moves to local minimal energy configurations. Continuous surface descriptions are created by extending the concept of two dimensional Delaunay triangulation to generate surface triangulations of arbitrary topology, embedded in three dimensional space.

1.5.2 Free-form Modeling

Our approach to free-form modeling provides the following abilities:

- to cut, merge, and join shapes,
- to remove and add material,
- to shape material using a small collection of tools,
- to add handles and loops without having to respecify the basic surface structure,
- to add curvature discontinuities by locally ignoring smoothness constraints,
- and to “heat” and “cool” the model to vary the material characteristics.

The modeling paradigm we present is based on a sculpting metaphor in which the user begins with a volume or sheet of material and arrives at a final shape through incremental sculpting operations In Figure 1.1 we show the result of deforming a sphere into a torus using two shaping tools. Such properties allow a designer to create a variety of geometric shapes without the need to be concerned with the underlying structure or parametrization of the shape.

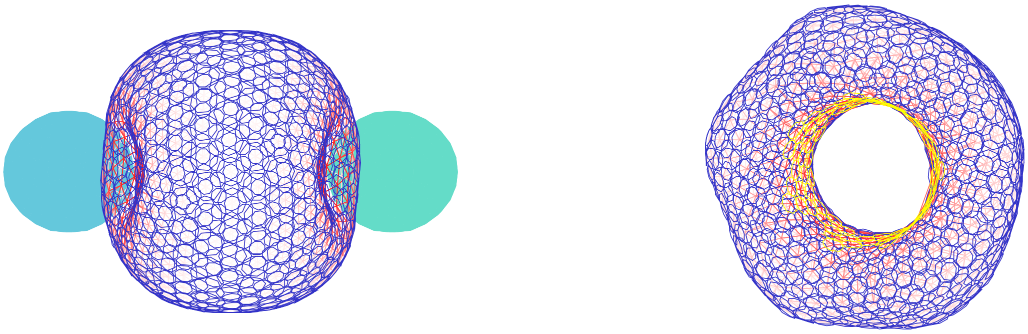


Figure 1.1: Free-form surface modeling

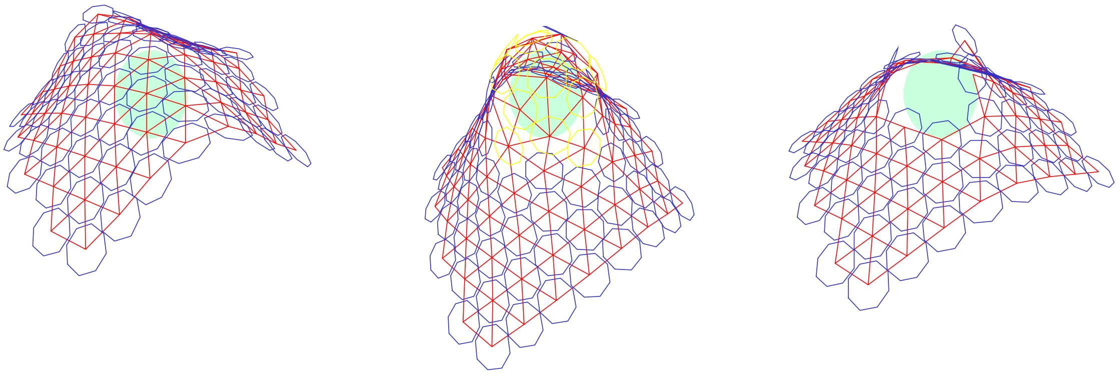
Deformation from sphere to torus using spherical shaping tools.

1.5.3 Computer Assisted Animation

When applied to three-dimensional animation, these same properties allow an animator to interactively sculpt the 3D models which can then be animated according to the associated physical characteristics. In addition, the physical characteristics may be modified during a scene to suit changes dictated by the story. For example, the technique can simulate from solid to fluid behavior, infinitely stretchable material, and materials that rip, tear, and that can be seamlessly merged together. Once created, the model may be influenced by a larger environment modeled for the animation. The manipulations can be global forces such as gravity, local forces derived from collisions with other objects, forces generated from hand gestures, scripted movement, or vector force fields defined in space. In Figure 1.2 we illustrate a selection of different physical properties, such as cloth draping (Figure 1.2a), plastic surface deformation (Figure 1.2b), and tearing (Figure 1.2c). The differences in physics were easily created by varying the weights of the potential energy functions.

1.5.4 Surface Reconstruction

When applied to surface fitting, the surface model and reconstruction methods developed retain the topological flexibility of the local patch methods, while constructing globally coherent surface models that can evolve consistently with time-varying data and forces. When reconstructing an object of arbitrary topology, the particles can be made to "flow" over the data, extracting and conforming to meaningful surfaces. The process is roughly analogous to pouring a viscous liquid over an object and wetting its surface. In Figure 1.3 we show the reconstruction of the surface from a CT scan of a plastic "phantom" vertebra model (decimated to $120 \times 128 \times 52$ resolution). 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



(a) (b) (c)

Figure 1.2: Computer assisted animation

Variable surface behavior: (a) cloth draping, (b) plastic deformation, and (c) tearing.

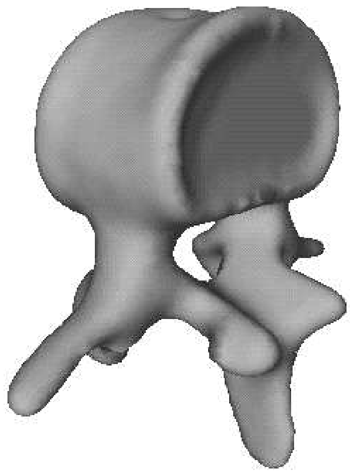


Figure 1.3: Surface reconstruction

Shaded 3-D Reconstruction of a vertebra from $120 \times 128 \times 52$ CT volume data.

values until a closed surface was obtained.

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

1.5.5 Results

The main results of this thesis include:

- A new surface model representation that we call "oriented particle systems". This model combines particle systems, differential geometry, and physics into a single model. The model uses potential energy functions for minimizing the normal curvature, Gaussian curvature, and the twist of a surface, for determining the principal directions of surface curvature and lines of curvature, and for emulating the stretching and bending energies of physical objects.

- A free-form modeling paradigm in which surfaces are sculpted using tools in analogy to sculpting with physical tools. Basic modeling operations include merging, cutting, extending surfaces, and specifying curvature discontinuities.

- A heat transfer model for dynamically coupled particles derived from the macroscopic heat equation.

- A method of varying the stiffness of Lennard-Jones coupled particle models based on thermal energy while maintaining a conservation of total system energy. Combined with the heat transfer model, this extends the range of behavior exhibited to be from solid to fluid. It also allows for local variations in malleability of a model thus providing sculptors more control over modeling operations.

- A triangulation algorithm for creating continuous surface descriptions of arbitrary topology from 3D point sets.

- An analysis of the computational and memory requirements of dynamically coupled particle systems. This includes an analysis of neighboring finding techniques and an analysis of the stability and accuracy of the explicit Leapfrog integration scheme for the Lennard-Jones potential.

- Surface reconstruction algorithms for both interpolating 3D point sets and optimal surface fitting of 3D data. The algorithms were designed for both reconstructing open and closed surface from sparse 3D point sets, polygonizing iso-surface functions, and segmenting 3D volumetric data, e.g. CT scan data.

- Visuals simulations of volumes and surfaces interacting with external objects and forces, free-form modeling, and surface reconstruction.

1.6 Thesis Overview

In Chapter 2 we review the background material and related work. In Chapter 3 we develop a spatially coupled particle system model of deformable volumes. In Chapter 4 we extend the model of Chapter 3 to "oriented particles" which prefer to arrange into surfaces rather than volumes. In Chapter 5 we provide mappings from the discrete nature of a particle system to continuous surface descriptions. In Chapter 6 we further extend the behavior of the surface and volume models to be functions of thermal energy. In Chapter 7 we discuss implementation and efficiency issues. In Chapter 8 we apply our model to the applications of physics based animation, interactive free-form modeling of surfaces, and surface reconstruction of 3D data. In Chapter 9 we conclude with a summary and final remarks.

In Appendix A we review the differential geometry of surfaces. In Appendix B we review Newtonian dynamics. In Appendix C we derive equations for computing the gradient of energy functions based on the Euclidean norm, products of the norm with normal vectors, and weighting functions. In Appendix D we derive forces and torques for the co-planarity, co-normality, co-circularity, and Lennard-Jones potential functions. In Appendix E we provide a finite element analysis of the surface energies.

Chapter 2

Background

2.1 Surface Modeling

There exist a variety of techniques to solve the shape representation and manipulation problem for specific sets of shapes. For example, piecewise spline patches have gained wide spread acceptance in applications where controlling the degree of surface continuity is important. Constructive solid geometry is used in computer aided design and manufacturing (CAD/CAM) to create solid volume descriptions of shape, where the intrinsic properties of a solid object such as a continuous inside/outside boundary, a closed surface, and volume constraints are important. Surfaces of revolution and extruded two dimensional outlines create shapes with symmetry along a given dimension. Any faceted surface can be modeled using polygonal meshes, and subsequently smoothed using subdivision methods. Implicit surfaces are easily defined as an iso-value of a continuous scalar three dimensional field, and are easily manipulated when the field is a function of a skeleton or set of control points. Free-form deformations and parametric warping of space allow shapes to be deformed at global and local levels. This section reviews the most common shape representations, and discusses their advantages and limitations.

2.1.1 Polygonal Meshes

Polygonal meshes are perhaps the most widely used shape representation in computer graphics. Geometrically this is the simplest shape representation technique that allows the description of a wide variety of shapes and topologies. It is a natural description for surfaces, and with appropriate, constraints a description for solid volumes. Due to its simplicity it is included in virtually every modeling system, creating a lowest common denominator for shape representation.

However, simplicity comes at a cost. Polygonal meshes are limited to accurately describing surfaces composed of planar facets and thus are unable to accurately represent curved surfaces. Instead, curved surfaces are approximated by polygons which linearly interpolate between points on the original surface. At a fine resolution, the visual artifacts introduced by such approximations are negligible for rendering applications. At lower resolution, interpolating the vertex normals across the area of the polygon during the rendering phase results in the illusion of a smooth surface. For other more demanding applications, the exact representation of curves are required and the polyhedral model falls short.

One solution to generating smooth surfaces is to apply surface subdivision methods. Subdivision can be used to recursively approximate a polygonal mesh with finer polygonal meshes, which in the limit results in a smooth surface (Catmull and Clark, 1978; Peters and Reif, 1997) except at a number of extraordinary points (Doo and Sabin, 1978). As an alternative to approximating, one can use an interpolating subdivision scheme which in the limit interpolates a smooth surface between the original polyhedral vertices (Zorin, Schröder and Sweldens, 1996).

Still, to sculpt a shape, a designer must specify the location of each vertex, the edges joining each vertex, and the series of edges and the order of edges belonging to each polygon. In addition, some systems require that the outward facing polygons all have the same "sign", that is the sign of the plane normal defined by traversing edges around the polygon, either clockwise or counterclockwise, must be the same. Even with interactive tools to help specify the vertices and connections, this is a tedious and time consuming process, especially for models containing thousands of polygons.

2.1.2 Parametric Representations

In a parametric representation, a shape is defined by a set of parameterized functions, such as

$$x(u,v), \quad y(u,v), \quad z(u,v),$$

where u and v are parameters, and a surface point $\mathbf{x} = (x(u,v), y(u,v), z(u,v))$ is given by evaluating the parameterized functions. The parametric representation has two distinct advantages. First, an arbitrary number of surface points are easily generated by sweeping the parameters u and v through their domain, thus facilitating the rendering process. Second, different levels of curvature continuity can be controlled by careful selection of the underlying parametric equations.

Shapes can be defined using either global parametric or piecewise parametric patches. Quadrics and superquadrics (Barr, 1981) are examples of global parameterized primitives used to define complete surfaces such as spheres, ellipsoids, and tori. For a wider range of parametric shapes, one usually applies a piecewise surface construction approach.

Traditional spline techniques (Bartels, Beatty and Barsky, 1987; Farin, 1992) 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, a surface's shape can be expressed using a superposition of basis functions

$$\mathbf{s}(u_1, u_2) = \sum_i \mathbf{v}_i B_i(u_1, u_2), \tag{2.1}$$

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 created by interactively positioning the control vertices or by directly manipulating points on an existing surface (Bartels and Beatty, 1989). Areas of a surface can be locally refined using a hierarchy of tensor-product B-splines (Forsey and Bartels, 1988).

2.1.3 Constructive Solid Geometry

Many applications, such as engineering and product design, require the representation of solid volumes. In constructive solid geometry (CSG), a solid is represented as a set-theoretic Boolean expression of primitive solid objects (Hoffman, 1989). Standard primitives are parallelepipeds (blocks), triangular prisms, spheres, cylinders, cones, and tori. Through a combination of simple primitives, complex shapes are easily constructed. The advantage of the CSG representation is that valid volumes can always be guaranteed.

To construct a shape, a user begins by instantiating a generic primitive by specifying the parameters of the primitive, such as the length and width of a parallelepiped or the radius of a sphere. Once instantiated, these primitives are combined using rigid motions and regularized Boolean set operations. These operations are regularized union, regularized intersection, and regularized difference. They differ from the standard set-theoretic operations by operating on the interior of the two solids, thereby eliminating lower dimensional geometric primitives which do not bound the resulting volume. The volume resulting from a regularized Boolean operation can then be combined with other volumes, until the final shape is realized.

2.1.4 Implicit Representation

In the implicit representation, a shape is defined as the locus of points that satisfy an equation $f(\mathbf{x}) = 0$, where $\mathbf{x}$ is a three dimensional point. For points inside of the surface, $f(\mathbf{x}) > 0$, and for points outside of the surface, $f(\mathbf{x}) < 0$.

Implicit surfaces enjoy several benefits such as the ability to efficiently compute inside/outside tests and the ability to easily build up complex shapes (Bloomenthal, 1989). For example, the standard set operations of union and intersection are easily implemented. Surfaces can be defined directly from a function or constrained in term of other geometric primitives. *Skeletal* surfaces can be defined in terms of distance constraints from a geometric entity. For example, a sphere is defined as a fixed distance from a point and a rounded cylinder is defined as a fixed distance from a line segment. Alternatively an implicit surface can be constrained to be a fixed distance from another surface, creating an *offset* implicit surface. The implicit formulation also allows for the blending of surfaces at branch points, a difficult problem for piecewise parametric surfaces. Implicit surfaces are often defined by combining algebraic functions based on control points, thus allowing surfaces to be easily deformed by displacing the control points (Blinn, 1982; Wyvill, McPheeters and Wyvill, 1986a).

A considerable disadvantage of the implicit formulation is that, in general, surface points cannot be directly computed as in the parametric representation. Algebraic surfaces may be ray-traced or rendered using incremental scan line techniques, and though certain class of implicit surfaces can be ray-traced (Blinn, 1982; Tonnesen, 1989; Wyvill and Trotman, 1989), no similar incremental techniques exist for arbitrary implicit surfaces (Bloomenthal, 1989). As an alternative, the implicit surface can be converted to a polygonal representation which can then be rendered with a conventional polygon renderer (Bloomenthal, 1988; Bloomenthal, 1989; Velho, 1990; Witkin and Heckbert, 1994). These techniques are discussed in more detail in Chapter 5.

2.1.5 Geometric Deformations

Geometric deformations provide another method for modeling shape. Mapping from one R^3 space domain to another R^3 space provides a "warping of space" and the geometry within that space, creating smooth deformations. By applying such a mapping Barr (1984) has shown the ability to bend, twist, and taper geometric objects. The normal and tangent vectors of the deformed object, as well as the ratio of volume change, can be calculated from the Jacobian matrix of the point transformation function. Deformations can also be realized by embedding a geometric object within a lattice of trivariate polynomials (Sederberg and Parry, 1986; Coquillart, 1990). By moving the control points of the lattice, one changes the R^3 to R^3 mapping and the embedded geometry is deformed. With appropriate constraints, the deformations can be defined both globally and locally, in a piecewise manner, and layered into a hierarchy of deformations. For more intuitive control the user can directly move points on the original surface to the desired deformed position (Hsu, Hughes and Kaufman, 1992), and then a least squares minimization is used to calculate the new lattice control point positions, thereby completing the mapping. While such mappings are very powerful, they preserve the underlying structure and parametrization of the surface, and thus do not allow the user to change topologies.

2.1.6 Variational Surfaces

By specifying "character lines", one can outline the shape of a surface and use either finite element (Celniker and Gossard, 1991) or variational techniques (Moreton and Séquin, 1992; Welch and Witkin, 1992; Welch and Witkin, 1994) to fit smooth surfaces between the character lines. The finite element approach provides physically realistic surfaces by minimizing an energy functional that describes the resistance to stretching and bending. Control of the final surface shape is achieved by first parameterizing the shape, then applying loads and geometric constraints. Terzopoulos and Qin (1994; 1996) use finite element techniques to solve a physics based generalization of non-uniform rational B-splines. Their model allows designers to sculpt shapes by applying forces and shape constraints, in addition to the traditional method of adjusting control points. The variational approach also uses geometric constraints and optimizes a constrained surface functional to create smooth surfaces. By stitching curves together the user can construct smooth shapes of arbitrary topology. Moreton and Séquin (1992) perform a non-linear optimization to minimize a fairness functional of the squared magnitude of the variation in principal curvatures. The result is a patchwork of Bézier patches which form a G^1 continuous surface. The variational surfaces of Welch and Witkin (1992) differ from the finite element and physically based models in that they "do not require the surface to respond in an intuitive or natural way to direct control-point manipulation". The variational surfaces of Welch and Witkin (1994) minimize the squared magnitude of the principal curvatures and are similar to implicit surfaces in that they allow the construction of arbitrary topology surfaces and, in general, cannot be explicitly computed.

2.1.7 Discussion

Existing surface representation techniques allow users to specify any geometric shape and any topology. However the specification of the shapes may be both time-consuming and tedious. Polygon meshes can be used to represent any faceted shape, and linearly approximate any curved surface. Piecewise parametric functions, such as spline surface patches, can be used to represent any curved surface with tangent plane or higher continuity conditions. Since polygons can be written as a parameterized equation, both polygonal meshes and parameterized surfaces are easily rendered by incrementally varying the parameters of the respective equations, thereby generating a stream of surface points. Implicit surfaces also allow the generation of geometric shapes of arbitrary topology, but do not possess a surface parametrization making it more difficult to render. Solutions to rendering are found through ray-casting techniques or by approximating with polygonal meshes or particles. However implicit surfaces have other valuable properties such as the guarantee of a closed surface, the ability to easily perform inside/outside tests, and the automatic blending of surfaces.

From a designer's point of view, perhaps the most serious drawback is the difficulty one has in creating and manipulating these complex shapes. The use of CSG allows the designer to add and subtract shapes with the knowledge that the final shape will always represent a closed solid volume. The use of free-form deformations allows shapes of static topology to be deformed both locally and globally. Finite element and variational techniques solve portions of the shape construction and manipulation problem by allowing the user to outline the shape as a set of character lines which the user then "skins" with a surface, which may then be locally deformed. These techniques address the construction of a valid geometric shape, but do not address the need to be able to manipulate the final shape at both the local and global level. While in some applications the specification of a shape is all that is needed, other applications such as animation require the ability to continually change the basic shape and in some cases even the topology, with minimal user intervention. The technique presented in this dissertation provides users a means to interactively shape surfaces, modify the surfaces locally and globally, including topological changes, with minimal user interaction.

2.2 Surface Reconstruction

Many vision researchers have investigated the reconstruction of $2\,{}^1\!/_2$-D viewer-centered surface representations (Terzopoulos, 1984; Boult and Kender, 1986). These representations are typically based on parametric spline models with internal strain energies. Equally intense effort has gone into the development of 3-D object-centered surface representations. These include generalized cylinders (Agin and Binford, 1973; Nevatia and Binford, 1977), superquadrics (Pentland, 1986; Solina and Bajcsy, 1990), and triangular meshes (Boissonnat, 1984), as well as their physics-based generalizations, dynamic deformable cylinders (Terzopoulos, Witkin and Kass, 1988), spheres (Miller et al., 1991; McInerney and Terzopoulos, 1993), superquadrics (Terzopoulos and Metaxas, 1991), and meshes (Vasilescu and Terzopoulos, 1992). Physically based models incorporate internal deformation energies and can be fitted through external forces to visual data such as 2-D images or 3-D range points. The $2\,{}^1\!/_2$-D viewer centered and 3-D object centered representations both assume a given topology, usually parameterized over a planar or spherical domain.

A common way to cope with unknown topological structure is to resort to a "patchwork" surface representation (Sander and Zucker, 1990) which abandons a global representation and describes the surface only locally in terms of planar, quadric, or cubic patches. A drawback of such local surface representations compared to globally parameterized geometric models is that they do not facilitate common surface analysis tasks such as area, curvature, and enclosed volume computations. More serious difficulties arise in the dynamic analysis of objects. Possible scenarios include the incremental reconstruction of surfaces from sequential views around objects, or the reconstruction, tracking, and motion estimation of dynamic non-rigid objects such as a beating heart. A globally consistent surface model can provide powerful constraints for solving these dynamic estimation problems.

Another approach to inferring topological structure is to construct a graph over the sample points which reflects spatially adjacent points (Hoppe et al., 1992; Edelsbrunner and Mücke, 1994; Guo, Menon and Willette, 1997). If the data are not sampled isotropically, that is with the same density in each dimension, then the correct surface may not be realized (Hoppe et al., 1992). The alpha shapes of Edelsbrunner and Mücke (1994) encode the spatial proximity relationship of the point set as a *simplicial complex*[1] overcoming the isotropic sampling restriction. In general, the result is not a connected surface interpolating the sample points, but a collection of points, lines, and surface patches with which the user is left to infer the shape. In a post-processing phase, the exterior faces of an alpha shape can be converted into a 2D-manifold surface (Guo, Menon and Willette, 1997), though due to the chosen alpha parameter the exterior faces may not adequately represent the full set of sample points, resulting in a loss of detail. To extract surfaces from medical images, *T-surfaces* (McInerney and Terzopoulos, 1997) overcome the topology restriction by continually recomputing a deformable surface based on an inside-outside classification

[1] A three-dimensional simplicial complex is a collection of k-simplices, $0 \leq k \leq 3$: A 0-simplex is a point, a 1-simplex is a line connecting two points, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron.

of a grid of voxels, analogous to the polygonization of implicit surfaces. A summary of deformable models applied to medical image analysis can be found in (McInerney and Terzopoulos, 1996).

Existing surface representations have limitations— viewer-centered methods make no attempt to represent non-visible portions of object surfaces, while object-centered methods make strong assumptions about object topology, and graph-theoretic methods make strong assumptions about the sampling density. This dissertation proposes a new approach to surface modeling which overcomes these limitations. The approach leads to flexible reconstruction algorithms which are able to compute detailed geometric descriptions that are not only inherently viewpoint invariant, but more importantly, are sufficiently powerful to represent surfaces of arbitrary topology. The algorithms can interpolate regular or scattered 3-D data acquired from an imaged object, without any a priori knowledge of the object topology.

2.3 Physically Based Modeling

2.3.1 Deformable models

Physically based modeling provides the ability to generate animations of physical phenomena through simulation. Starting with a parametric representation for the surface $\mathbf{s}(u_1, u_2)$ and adding a physical level of abstraction, Terzopoulos *et al.* (1987) create elastically deformable surfaces. To define the dynamics of the surface, they use weighted combinations of different tensor (stretching and bending) measures to define a 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 movement of a 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 geometric spline model.

Physically-based surface models have been used to model a wide variety of materials, including cloth (Breen, House and Getto, 1991; Terzopoulos and Fleischer, 1988b), membranes (Terzopoulos et al., 1987), and paper (Terzopoulos and Fleischer, 1988b). Viscoelasticity, plasticity, and fracture have been incorporated to widen the range of modeled phenomena (Terzopoulos and Fleischer, 1988b). By adding muscles and skin to an otherwise inert model, the movement of characters, such as worms (Miller, 1988), fish (Tu and Terzopoulos, 1994), and human faces (Waters and Terzopoulos, 1990; Lee, Terzopoulos and Waters, 1995), can be automatically generated.

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, Witkin and Kass, 1987). 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 case, defining the model topology in advance remains a tedious process. Furthermore, it severely limits the flexibility of a given surface model.

2.3.2 Fluid Models

The complex nature of liquids is extremely fascinating and poses a number of problems for research in computer graphics. Liquids exhibit a wide range of phenomena, such as conforming to the shape of containers, wave propagation, cresting and breaking as exhibited by ocean waves, splashing, sheeting, foam, and bubbles. Capturing all of these in one model is difficult and hence a variety of techniques have been proposed, each modeling a subset of the phenomena. Trigonometric functions have been used to model ocean waves (Peachy, 1986; Fournier and Reeves, 1986). Individual particles have been used to model the spray of water from boat wakes (Goss, 1990), the spray of cresting waves (Peachy, 1986), splashes (O'Brien and Hodgins, 1995), and waterfalls (Sims, 1989). Three approaches to modeling water as height fields have been proposed: as a linearized approximation to the shallow water equation (Kass and Miller, 1990), as columns of fluid where the volumes of fluid transferred between columns is conserved (O'Brien and Hodgins, 1995; Mould and Yang, 1997), and a solution of Navier-Stokes equations in two dimensions that is then mapped to a 3D height field (Chen and Lobo, 1995). To model rapid changes in topology of viscous liquids, coupled particle systems with attractive-repulsive forces have been used (Miller and Pearce, 1989; Terzopoulos, Platt and Fleischer, 1989; Tonnesen, 1991; Desbrun and Gascuel, 1995; Reynolds, 1997) as well as smoothed particle hydrodynamics (Roy, 1995; Desbrun and Gascuel, 1996) (discussed in Section 2.4.1). The particle based models are discussed further in Section 2.4.2. Recent work has solved the Navier-Stokes equations over a low resolution regular grid using a 3D finite-difference approximation (Foster and Metaxas, 1996; Foster and Metaxas, 1997a; Foster and Metaxas, 1997b).

2.4 Particle Systems

A particle system is a collection of point masses with associated forces whose movement is governed by the laws of physics. To describe each particle, a set of attributes, such as mass, position, velocity, and acceleration, are assigned to the particle. Potentials are commonly used to generate forces acting on the particles, and the movement of the particles is given by the laws of Newtonian physics,

$$\frac{\mathbf{f}_i(t)}{m_i} = \frac{\mathrm{d}\mathbf{v}_i(t)}{\mathrm{d}t}; \qquad \mathbf{v}_i(t) = \frac{\mathrm{d}\mathbf{x}_i(t)}{\mathrm{d}t},$$

where $\mathbf{f}_i$, $\mathbf{v}_i$, $\mathbf{x}_i$, and m_i are the force acting on, the velocity, the position, and the mass of particle i. Given initial conditions, these systems can be simulated over time by integrating the equations of motion. Depending on the forces applied, such systems can model a variety of complex and time dependent behavior. Rather than modeling as an Eulerian dynamical system, where the system state is defined at fixed grid samples, particle systems use a Lagrangian approach, where the samples of state follow the movement of the system.

2.4.1 Particle Systems in the Physical Sciences

In the physical sciences, particle systems have been used by to model a variety of phenomena including the evolution of galaxies, plasma, the properties of semiconductors, magnetic fields, compressible gas flows, and the phase changes in matter (Hockney and Eastwood, 1988; Heyes, 1998; Monaghan, 1992). Typically, each particle in the system models a primitive element, such as a star or molecule, of the phenomena under study. To predict the correct dynamic behavior, such simulations require the computation of complex interactions with high numerical accuracy. This problem is aggravated by the number of elements in the systems under consideration. For example, on a microscopic scale, the number of molecules in an ounce of water is on the order of 10^{25}. And on a cosmological scale, the number of stars in a galaxy[2] is on the order of 10^{10} to 10^{12} (Hockney and Eastwood, 1981).

Astrophysicists model the evolution of star systems based on the density of bodies and gravitational fields (Hockney and Eastwood, 1988; Heggie, 1987). To maintain accuracy, several approaches have been suggested. When modeling small clusters of stars, on the order of a 1000, the system can be directly modeled as a particle system with each particle representing a star, and forces computed directly. This requires $O(N^2)$ operations for N particles. For coulombic interaction[3], the computations can be reduced to $O(N)$ by using particle-mesh methods where short range forces are computed directly between particles, and long range forces are computed over a mesh (Greengard, 1988; Zhao, 1987). For larger systems, each particle models the mean properties of density and gravitional fields for clusters of approximately 10^6 stars, thus allowing a system of 10^4 particles to model a galaxy.

Molecular dynamicists have used particle systems to study solids, liquids, gases and the phase changes between these states (Barton, 1974; Christy and Pytte, 1965; Temperley, 1978; Trevena, 1975; Heyes, 1998). The concept of a pairwise intermolecular potential energy function has proven valuable in describing inter-molecular interactions in a quantitative fashion. The short range repulsive forces can be modeled by a potential energy function represented as an exponential expression $\phi_R(r) \propto r^{-n}$, where r is the distance between the two molecules. The long range attractive forces, to a reasonable approximation, can be treated together as a single expression, and modeled by the potential function $\phi_A(r) \propto r^{-m}$. Higher body interactions (e.g. 3-body interactions) can be computed for molecular systems, though the effects of these interactions are usually incorporated into the model by modifying the values of a pairwise potential (Heyes, 1998). In addition to modeling the phase transitions, particle systems have also modeled the macroscopic properties of matter, such as

[2]The number of galaxies in the observable universe is estimated on the order of 10^9.

[3]Coulombic potentials are inversely proportional to distance, that is r^{-1}. Coulombic forces are proportional to r^{-2}.

temperature, volume, and local geometry. Similar to modeling galaxies, modeling large molecular systems with particles is computationally expensive. Simulations which ignore distant particle forces have been used to reduce the computational cost, although this approach has been found to contribute to sensitivities in the computed phase diagram, especially close to critical points (Heyes, 1998). In modeling the phase changes between states, physicists are in effect modeling changes in *structure*. It is the ability to provide such fluid changes in structure that we wish to capture in our approach to the volumetric modeling of deformable materials.

Smoothed particle hydrodynamics (SPH) is a Lagrangian method used to study fluid flow in astrophysics (Monaghan, 1982; Monaghan, 1985; Monaghan, 1988; Monaghan, 1992), in particular compressible fluids such as stellar gases. SPH is based on the mathematical identity

$$A(r) = \int A(s)\delta(r-s)ds$$

given here in one dimension, where δ is the Dirac delta function defined to be zero for all non-zero values of $(r-s)$. Any field $A(r)$ can be approximated by replacing the Dirac function with an interpolating kernel $w(u,h)$ of compact support, such that in the limit as the "smoothing length" h goes to zero, the kernel equals the delta function. In three dimensions a discrete approximation of the continuous integral is given by representing the volume as particles, and summing the kernel weighted contributions from each particle. This formulation allows the density of the fluid at any point in space to be approximated as a weighted sum of particle masses. Forces on particles result from solving the gradients in pressure as functions of the density at each particle. Over a set of uniformly ordered particles, the SPH method is equivalent to finite-difference schemes, with the particular scheme dependent on the choice of interpolating kernel. Error is minimal when the particles are equally spaced and increases as the particles become disordered.

2.4.2 Particle Systems in Computer Graphics

In computer graphics, particle systems have been used to model visually complex natural phenomena such as fire, foliage, and waterfalls; to model and reconstruct both surfaces and volumes; and to emulate the physics of deformable, elastic, viscous, and solid materials. To aid in the review of related work, we categorize particle systems according to the interactions between particles. In *systems of independent particles*, the forces on each particle are independent of other particles in the system. *Particle systems with fixed connections* interact with neighboring particles where the set of interactions is constant after the initial specification. In particle systems with *spatially coupled particle interactions*, the interactions between particles evolve over time due to their relative spatial state. This results in the ability to model both varying geometry and topology as will be shown in the thesis.

Systems of independent particles have been used to model *visually complex* natural phenomena such as fire, smoke, foliage, and the spray of splashing water (Reeves, 1983b; Reeves and Blau, 1985; Sims, 1990; Stam and Fiume, 1993; Stam and Fiume, 1995; O'Brien and Hodgins, 1994; Goss, 1990; Sims, 1992). In these systems, forces on each particle are independent of the other particles in the system. To create complex behavior, these techniques use large numbers of particles reacting to forces such as gravity, obstacles, wind fields, and turbulence. Particles are created and deleted from the system using rules based on the phenomena being modeled. Most of these approaches concentrate on creating a particular visual effect and make no attempt to define either an object volume[4] or the corresponding surface.

The previously mentioned deformable models of (Terzopoulos and Fleischer, 1988b; Haumann et al., 1991; Breen, House and Getto, 1991; Breen, House and Wozny, 1994) and the surface reconstruction model of (Miller et al., 1991) can be categorized as particle systems with fixed particle interactions. The forces felt by a particle are in part due to the fixed inter-particle connections and in part due to external forces. Shapes modeled by a system with fixed particle interactions can be deformed to change the surface geometry, but are limited to the structure imposed by the original connections.

In particle systems with spatially coupled particle interactions, the interactions between particles evolve over time — connections are automatically broken and *new* connections are automatically created. Replacing the fixed set of interactions with interactions that dynamically evolve creates a flexible modeling paradigm in which geometric and topological changes can occur as the underlying structure of the system changes. We briefly mention work cited early to place them in context of particle systems. The physically-based deformable volumes and fluids of (Miller and Pearce, 1989; Terzopoulos, Platt and Fleischer, 1989; Desbrun and Gascuel, 1995; Reynolds, 1997) are spatially coupled particle systems. The related volume models of Roy (Roy, 1995) and Desbrun and Gascuel (Desbrun and Gascuel, 1996) have applied Monaghan's SPH model (Monaghan, Thompson and Hourigan, 1994) of nearly incompressible fluids. To approximate surfaces, spatially coupled particle systems have been used to re-mesh polygonal models (Turk, 1992), to triangulate implicit surfaces (Witkin and Heckbert, 1994; Crossno and Angel, 1997) and variational surfaces (Welch and Witkin, 1994). Spatially coupled particle systems have also been used to distribute paint strokes for a painterly effect in rendering applications (Meier, 1996), and to grow cellular based textures (Fleischer et al., 1995).

2.4.3 Discussion

Sculpting with particle systems does not fall cleanly into previous modeling paradigms. It is neither a parametric representation, an implicit representation, nor a solid modeling primitive. However, we can construct such representations from our particle-based model. For the oriented particle system, a natural continuous surface representation is a triangulation interpolating the particle positions. From this, a subdivision surface (Loop, 1987) or smooth piecewise parametric representation (Loop, 1994) can be constructed. For the un-oriented particle system, an implicit surface is a natural

[4]An exception is (Stam and Fiume, 1993; Stam and Fiume, 1995) who use "blobs" to define a volume.

surface representation. As a sculpting metaphor, our work (Szeliski and Tonnesen, 1992) shares similarities to the recent work of (Welch and Witkin, 1994) and (Witkin and Heckbert, 1994) in that both techniques relying on a triangulation of a particle system to render the surface, though the origin of the surface is distinctly different. In (Welch and Witkin, 1994), particles are used not to define the surface, but to quickly approximate a variational surface. Instead of using a particle system as a basis for continual changes in topology, as this dissertation does, they add "just enough structure to the particle system to unambiguously fix its topology". Since their surfaces cannot be explicitly computed, they approximate them via a triangulation, using a particle system as a step in the triangulation process. In (Witkin and Heckbert, 1994) they also present a coupling between particles and an smooth surface, but in this case they use particles for both sampling and as control points of an implicit surface. The sampling of the surface using repulsion forces is similar to (Welch and Witkin, 1994) and (Turk, 1992). The computational expense of using particles as implicit surface control points requires over $O(N^3)$ time, where N is the number of control particles. We also allow particles to act as control points, but we have designed our system to execute in $O(N \log N)$ in the number of particles.

Lombardo and Puech (Lombardo and Puech, 1995) extended our oriented particles to endow objects with memory of their original shape. In this case, particles prefer a rest state matching originally specified curvature measures instead of a default zero curvature measure as our method does. This allows them to create oriented particle skeletons for modeling implicit surfaces with "shape memory".

In surface reconstruction, it is the dynamic nature of our model that is important to extracting the surface structure from a set of 3D data. By allowing our surfaces to extend out from known surface samples we can extend the range of surface reconstruction. For example, this will allow us to overcome some problems associated with the anisotropic sampling or random under-sampling in regions. In areas of high sampling density, the samples will be sufficient to reconstruct the correct surface, while the growth of new particles in low density sampled regions will approximate the under-sampled area. As adjacent surface patches meet the particles, patches will automatically join together, completing the interpolation. The energy based nature of the particle system allows us to optimally fit surfaces to data based on minimizing flatness and curvature functions. Our particles can be thought of as providing a finite difference solution to a minimization problem, similar to deformable models. The difference between this and an actual finite difference scheme is that our grid is not fixed, but dynamic in nature.

The nature of our particle system has several advantages for animation. The physical properties embedded in our model allow the animator to mimic a variety of common real-world materials, reducing the amount of animator effort required to create a sequence. Since the shape will be continually changing, we do not want to impose unnecessary constraints on the animator. We would also like to point out that animating a shape is closely related to sculpting a shape. In fact, animation could be considered a continuous sculpting exercise, with each new image representing a complete sculpture. We feel the flexibility our model provides for sculpting will be directly applicable to animation.

We apply particle systems to model volumes as do (Miller and Pearce, 1989; Terzopoulos, Platt and Fleischer, 1989), extending their attractive-repulsive inter-particle force model, based on the Lennard-Jones function, to create a thermoelastic model in which the stiffness varies as a function of thermal energy. This provides a mechanism by which the model can mimic the "melting" and "freezing" of objects. Two recent papers suggest alternative attractive-repulsive forces. Lombardo and Puech suggest a "cohesion" force which has a similar shape to the Lennard-Jones force that we use, and assert this force reduces oscillations, allowing the system to reach a rest state sooner. Reynolds (Reynolds, 1997) suggests using Boscovich's law of force which has multiple minimal energy states, which he states is better for modeling inelastic deformations.

The recent work of Roy (1995) and Desbrun and Gascuel (1996) model fluids using the smoothed particle hydrodynamics model (Section 2.4.1). In the SPH model, forces between particle pairs are a result of gradients in pressure over the volume. As particles approach, the density and hence pressure increase, resulting in repulsive forces. As particles separate, the density and pressure becomes lower than the surrounding areas, resulting in attractive forces which equalize pressure. In particular, one must be careful that particles do not become too close with respect to the smoothing length; otherwise the particles tend to "clump" together in an unrealistic fashion. This is an artifact of the gradient of the kernel and is discouraged by

- adding a velocity-based damping term (Monaghan, Thompson and Hourigan, 1994) which is analogous to the ideal viscous unit (3.18) that this thesis employs for modeling visco-elastic materials,

- lowering the smoothing length (Roy, 1995),

- or defining a cusp-shaped smoothing kernel (Desbrun and Gascuel, 1996) such that the magnitude of the derivative increases rather than decreases when approaching an inter-particle separation distance of zero.

A cusp-shaped kernel results in a force curve that is similar in shape to the Lennard-Jones force curve (Desbrun and Gascuel, 1996)[Figure 3]. It differs in that the curve converges to a constant slope for small separations, like an ideal spring does, rather than to infinity as the Lennard-Jones model does. Thus the fluids SPH models are more compressible than fluids modeled using the Lennard-Jones function.

All of the spatially coupled particle volume models share features in common:

- Each particle represents a small volume element.

- The equations defined over pairs of particles result in attractive and repulsive forces in the direction of the vector separating neighboring particles.

- They can model elastic and visco-elastic materials.

- They are inadequate at *accurately* modeling incompressible materials, such as liquids, but they can model *nearly incompressible* fluids.

- Stiffer, less compressible materials require smaller time steps than more compressible materials.

In summary, our model is based on previous particle systems work in computer graphics and inspired by the physical sciences. The particles interact according to pairwise potential energy functions. These potential energies, inspired by physics and differential geometry, share similarities to the energy functions used for deformable models and variational surfaces. The self-organizing nature of our system is distinctly different than the traditional modeling techniques where one manually specifies the connectivity of surface patches, as is done in spline, polygonal, and variational based surface modeling. It allows for the ability to join and separate objects as do implicit surface modeling and CSG methods. While our model is a point based sampling description rather than a continuous description, we can generate full surface descriptions. The surfaces generated are implicit surfaces for volumetric samplings and triangulated polygonal models for surface samplings. The polygonal models can then be converted to smooth surfaces using either surface subdivision techniques or triangular based splines. Our model, like deformable models, allows us to create animations of visco-elastic materials, e.g. cloth. In addition our synthetic materials can be stretched, ripped, and joined back together automatically. Our model shares similarities to previous particle based volume models and recent fluid based particle models. We can also construct viewpoint invariant 3D surfaces that interpolate sparse point data and fit optimal surfaces to 3D volumetric data. Unlike object centered methods, the surfaces can be of arbitrary topology and genus without requiring prior assumptions of the surface structure.

Chapter 3

Particle Volumes

In order to develop a new model for shape representation and manipulation, we construct synthetic materials that exhibit useful physical and geometric properties. We do not try to simulate a given material accurately, but rather to imitate properties that we find useful for sculpting, animation, and reconstruction tasks. The ability to create large changes in the geometric structure and the ability to change the topology of the shape at will, suggests the use of primitive shape elements, where individual elements do not enforce a given topology, but rather the topology is an emergent property of the the self-organizing system. As such, we have chosen to use an interacting particle system that moves in accordance with the laws of Newtonian dynamics. The physical properties of our system follows from our selection of inter-particle potential energy functions. The geometry is derived from the relative positions of the particles.

3.1 Particle Systems

Each particle represents a primitive element with mass, volume, and physical properties defined between particles. A particle system is defined as a collection of point masses, where each particle has a position and mass, and moves under the influences of forces according to the principles of classical physics. Such a system is governed by the set of ordinary differential equations of motion

$$m_i\ddot{\mathbf{x}}_i + \gamma_i\dot{\mathbf{x}}_i + \mathbf{f}_i^{\mathtt{int}} = \mathbf{f}_i^{\mathtt{ext}} \qquad i = 1, \ldots, N, \tag{3.1}$$

where the subscript i denotes an attribute of particle i, N is the number of particles in the system, over struck dots denote time derivatives, m_i is its mass, $\ddot{\mathbf{x}}_i$ is its acceleration, $\dot{\mathbf{x}}_i$ is its velocity, γ_i is a damping coefficient that controls the rate of dissipation of the particle's kinetic energy, $\mathbf{f}_i^{\mathtt{int}}$ is the sum of inter-particle forces and $\mathbf{f}_i^{\mathtt{ext}}$ is the sum of external forces acting on the particle. Inter-particle force terms are functions of the form

$$\mathbf{f}_i^{\mathtt{int}}(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N) \tag{3.2}$$

External forces are functions of single particle state and external state variables

$$\mathbf{f}_i^{\mathtt{ext}}(\mathbf{x}_i, \mathbf{S}), \tag{3.3}$$

where $\mathbf{S}$ is a set of external state variables such as gravity, the positions of obstacles or shaping tools, and volumetric data sets. A particle system can be tailored to a specific application by choosing appropriate inter-particle and external forces.

3.1.1 Potential Energy

Potential energy functions provide an elegant method of describing inter-particle force functions based on particle positions. For a potential function ϕ, the force exerted on particle i with position $\mathbf{x}_i$,

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i}\phi$$

is due to the gradient of the potential energy ϕ

$$\nabla_{\mathbf{x}_i}\phi = \left(\frac{\partial\phi}{\partial\mathbf{x}_{1_i}}, \frac{\partial\phi}{\partial\mathbf{x}_{2_i}}, \frac{\partial\phi}{\partial\mathbf{x}_{3_i}}\right)$$

with respect to the change in position.

A common practice, and one this dissertation adopts, is to define a particle's potential energy based on the pairwise additivity assumption. This assumption states that the total potential energy of a particle is the sum of the pairwise potential energies between that particle and every other particle. That is, the potential energy ϕ_i of particle i with respect to a system of N particles is given by

$$\phi_i = \sum_{j\neq i}^{N} \phi_{ij}, \tag{3.4}$$

where ϕ_{ij} is the potential energy between particles i and j, and N is the number of particles in the system. Thus the net inter-particle forces acting on a particle i due to the potential energy function is

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i}\phi_i = -\sum_{j\neq i}^{N} \nabla_{\mathbf{x}_i}\phi_{ij}. \tag{3.5}$$

The total potential energy of the system is the sum of the pairwise potential energies for all pairs of particles

$$E_P = \sum_{i}^{N} \phi_i = \sum_{i}^{N}\sum_{j\neq i}^{N} \phi_{ij}. \tag{3.6}$$

3.1.2 Kinetic Energy

The kinetic energy of the system is a measure of particle movement. The kinetic energy K_i of a single particle i is given by

$$K_i = \frac{1}{2}m_i v_i^2 = \frac{1}{2}m_i\|\dot{\mathbf{x}}_i\| \tag{3.7}$$

where m_i is the mass of the particle, and v_i is the scalar speed of the particle. The kinetic energy of the system is the sum of the individual particle kinetic energies

$$E_K = \sum_i^N K_i. \tag{3.8}$$

3.1.3 System Energy

The total energy of the system, E_S, is simply a summation of the individual particles' kinetic and potential energies,

$$E_S = E_K + E_P = \sum_i^N K_i + \sum_i^N \sum_{j \neq i}^N \phi_{ij}. \tag{3.9}$$

Systems whose dynamics are governed by potential functions and damping will evolve towards lower energy states. Differentiating the potential energy functions results in forces acting on the particles. These forces move the particles over time resulting in an increase in kinetic energy. Assuming no external forces, eventually the system will come to rest as the kinetic energy is dissipated by velocity based damping.

3.2 Dynamic Coupling

The equations of potential energy from the previous section provide a generalized framework from which to work, yet for practical purposes they are problematic. To compute all inter-particle forces, without any restricting assumptions on the potential energies, requires N^2 force computations. For a system with thousands of particles, this is clearly a problem. Since we are not concerned with the long range effects between particles, as astrophysicists are when studying the evolution of a galaxy, we instead choose to limit the inter-particle interactions so that only neighboring particles interact. We define neighboring particles to be pairs of particles that are closer than a specified neighborhood distance. Thus we can reduce force computations to $O(N \log N)$ for neighbor finding and $O(N)$ to compute the forces. In Chapter 7 we discuss in more detail the choice to limit particle interactions, how we compute the nearest neighbors, and how we derive the complexity measures.

Letting $\mathcal{N}_i$ be the set of particles that neighbor particle i, we can rewrite the force and energy equations as follows. Particle i's inter-particle potential energy (3.4) is rewritten as

$$\phi_i = \sum_{j \in \mathcal{N}_i}^N \phi_{ij}, \tag{3.10}$$

and the corresponding inter-particle force (3.5) is rewritten as

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i} \phi_i = -\sum_{j \in \mathcal{N}_i}^N \nabla_{\mathbf{x}_i} \phi_{ij}. \tag{3.11}$$

The kinetic energy equations (3.7) and (3.8) are unchanged. The total potential energy of the system (3.6) and the total system energy (3.9) can be rewritten in a similar manner, though we omit them for brevity.

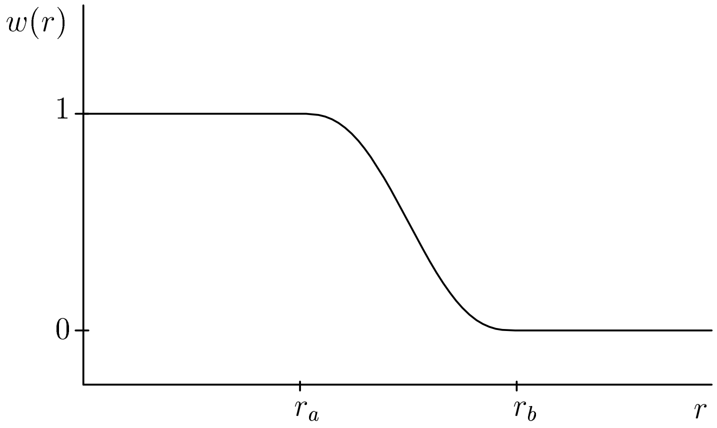


Figure 3.1: Potential weighting function.

3.2.1 Weighting Function

Ignoring distant neighbors, as described above, is sufficient under certain circumstances, but one must be careful. If at the neighborhood boundary, a given particle's contribution to the inter-particle force calculations result in a large enough discontinuity, then undesirable artifacts, such as instabilities in the numerical integration and visual artifacts, may occur. We present a solution to this problem now, and defer a detailed discussion to Chapter 7. Instead of ignoring particles outside of the neighborhood range, a better solution is to insure the inter-particle potentials, and hence forces, tend to zero at the neighborhood boundary.

We can enforce this condition by appropriately weighting the potential. To maintain the nature of the original potential, desirable properties of such a weighting function are that it be: monotonically decreasing from unity to zero; continuous and smooth; and continuous and smooth in the first and second derivatives. The smoothness and continuity conditions are important for well behaved numerical integration. To meet these conditions, we designed the following piecewise continuous function,

$$w(r) = \begin{cases} 1 & \text{if } r < r_a \\ g(s) & \text{if } r_a \leq r \leq r_b, \text{ letting } s = \frac{r-r_a}{r_b-r_a} \\ 0 & \text{if } r > r_b. \end{cases} , \tag{3.12}$$

where r is the distance between two particles, and $0 < r_a < r_b$.

We implement g as a fifth degree interpolating polynomial

$$g(s) = -6s^5 + 15s^4 - 10s^3 + 1.0.$$

over the interval $[0, 1]$. We designed the polynomial such that the first and second derivatives of $g(s)$ are zero for $s = 0$ and $s = 1$. Figure 3.1 shows the graph of $w(r)$. We generally set r_a to be the standard spacing between particles and r_b to be equal to the neighborhood boundary range. In essence, the weighted potentials have compact support. Using such a weighting function we insure that summing weighted potentials over all particles is equivalent to summing weighted potentials over nearest neighbors, that is,

$$\sum_{j \neq i}^{N} w(r_{ij})\phi(r_{ij}) = \sum_{j \in \mathcal{N}_i}^{N} w(r_{ij})\phi(r_{ij})$$

where $r_{ij} = \|\mathbf{r}_{ij}\| = \|\mathbf{x}_j - \mathbf{x}_i\|$ is the distance between particles i and j.

Having redefined the potential with a weighting function, the corresponding force is also redefined. The new force is given by evaluating the gradient with respect to the position

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i} \left(w(r_{ij})\phi(r_{ij}) \right) = \hat{\mathbf{r}}_{ij} \left(w \frac{\mathrm{d}\phi(r_{ij})}{\mathrm{d}r} + \phi \frac{\mathrm{d}w(r_{ij})}{\mathrm{d}r} \right), \tag{3.13}$$

where $\hat{\mathbf{r}}_{ij}$ is the unit vector in the direction of $\mathbf{r}_{ij}$, that is

$$\hat{\mathbf{r}}_{ij} = \frac{\mathbf{r}_{ij}}{\|\mathbf{r}_{ij}\|}.$$

The derivation of (3.13) can be found in Appendix C.6.

3.3 Creating Deformable Volumes

Deformable solids inspired by finite element theory have been modeled using hexahedral assemblies of point masses, springs, and damping elements (Terzopoulos and Fleischer, 1988a). In such spring-mass models the springs are *structural elements* that hold the object together. As two particles connected by a spring are separated, the force exerted by the spring steadily increases, pulling the point masses back together. To model a material that fractures, one can "break" a spring when it is stretched beyond a threshold distance. In general, spring-mass systems are good for modeling solids with fixed structure exhibiting small deformations and fracturing. It is not an adequate representation for modeling materials exhibiting large geometric deformations or changes in genus. Instead of a spring potential energy that encourages particles to maintain a fixed structure, we would like a potential energy that allows groups of particles to be separated and joined back together in new and different configurations.

Our goal is to provide an alternative model that allows for large changes in geometry, topology, and genus. To do so, we use a inter-particle potential that is elastic for small deformations, yet allowing for the rearrangement of elements over large deformations. To allow for rearrangement, without manually redefining the connections, the function is defined over all particle pairs, instead of a fixed set of pairs as in mass-spring systems.

3.3.1 Lennard-Jones Potential

The Lennard-Jones potential energy function fulfills these criteria (Heyes, 1998). It creates long-range attractive forces and short range repulsive forces which encourage

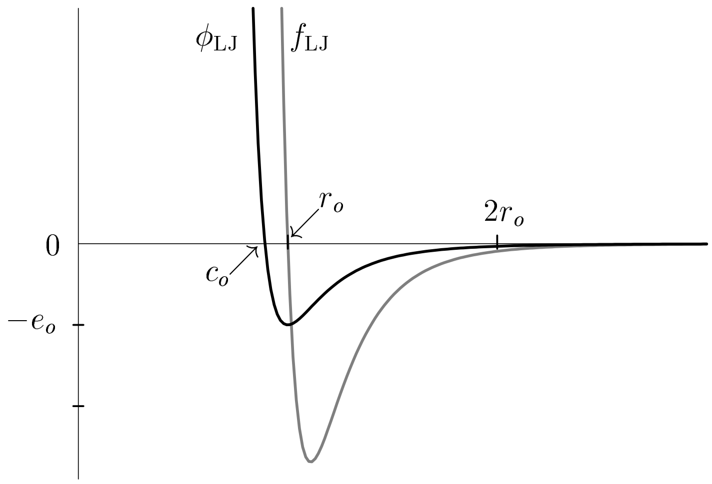


Figure 3.2: 12-6 Lennard-Jones function

The potential function for $n = 12$ and $m = 6$ is shown in black and the corresponding force shown in gray. The collision distance c_o, the equilibrium separation r_o, and the dissociation energy e_o are all labeled on the graph.

---

particles to maintain equal spacing. It also allows particles to be rearranged relative to one another, and yet does not require the manual specification of inter-particle connections. The Lennard-Jones energy function is defined as a function of separation distance r between a pair of particles

$$\phi_{LJ}(r) = \frac{B}{r^n} - \frac{A}{r^m}. \tag{3.14}$$

In Figure 3.2, we show a Lennard-Jones function with $n = 12$ and $m = 6$, as typically used in the molecular dynamics literature. When two particles are in equilibrium, the potential energy between them is minimal and marked in Figure 3.2 at $-e_o$. The magnitude of this energy is known as the *dissociation energy* and is the energy required to completely separate two particles. The distance between two particles when in equilibrium is known as the *equilibrium separation distance*, r_o. The Lennard-Jones potential goes to zero at two points, at infinity and at a distance defined as the *collision distance*, c_o. These three quantities (e_o, r_o, c_o) are labeled in the figure.

An alternate formulation, called the Lennard-Jones bi-reciprocal function

$$\phi(r) = \frac{-e_o}{m-n}\left(m\left(\frac{r_o}{r}\right)^n - n\left(\frac{r_o}{r}\right)^m\right), \tag{3.15}$$

provides a convenient method of tailoring the potential function to a specific equilibrium separation and dissociation energy. This formulation is equivalent to (3.14) with

$$A = \frac{-e_o n {r_o}^m}{m-n} \qquad B = \frac{-e_o m {r_o}^n}{m-n}. \tag{3.16}$$

In this form it is easy to see that the collision distance is a function of the equilibrium separation r_o and exponents n and m. For the case of $n = 2m$ it is

$$c_o = r_o(2^{-1/m}). \tag{3.17}$$

Instead of using the typical molecular dynamic values of n, m, and r_o which correlate to the forces felt at the molecular level, we choose values which more closely mimic behavior found at the macroscopic level. At the macroscopic level a first order approximation of deformation can be modeled using a spring potential

$$\phi(r) = k(r - r_o)^2$$

between point masses (Terzopoulos and Fleischer, 1988a). In Figure 3.3 we compare the spring potential to the weighted and unweighted Lennard-Jones potential with $n = 4$ and $m = 2$. We define the constants r_a and r_b of the weighting function to be equal to r_o and $2r_o$ respectively. Close to the equilibrium separation, the Lennard-Jones function has a parabolic shaped potential energy well similar to that of the spring potential energy function.

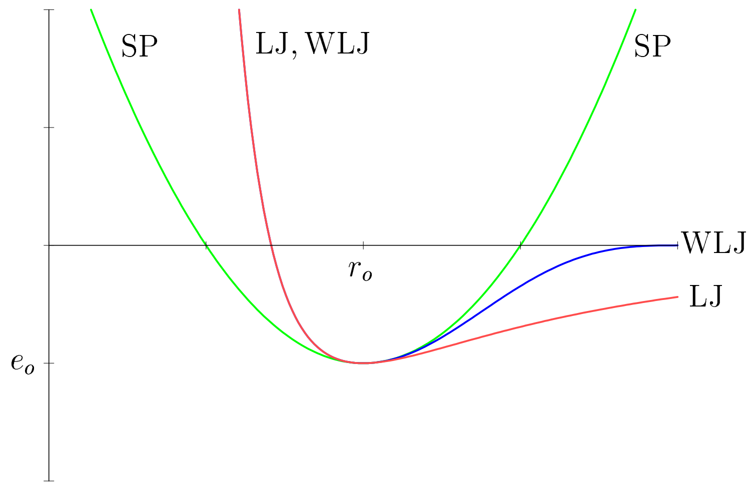
Similar to increasing the value k to increase stiffness in the spring model, we can increase the dissociation energy to increase stiffness in the Lennard-Jones particle model. Varying the exponents n and m varies the width of the potential well while keeping the minimum potential energy constant. Lower values result in wider potentials and more compressible materials, while higher values result in thinner potential wells and less compressible materials. A wide potential well will also result in flexible materials while a thin potential well will result in more rigid materials. Like large exponents, large dissociation energies will result in large forces and thus more rigid materials. The difference is incompressibility (the degree of volume preservation) and how "brittle" the material is. For equivalent forces, brittle materials (high exponents, low dissociation energy), require less energy to break the inter-particle bonds than non-brittle materials (low exponents, high dissociation energy).

3.3.2 Damping

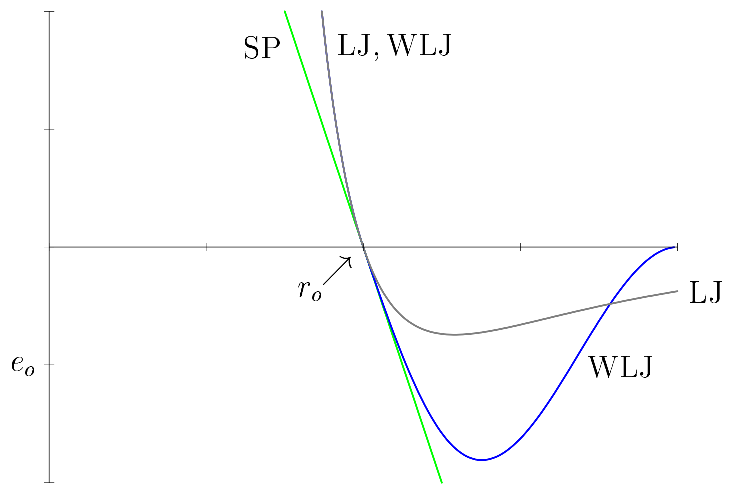
The second term of (3.1) is a velocity based damping force, $-\gamma_i \dot{\mathbf{x}}_i$. When $\gamma_i > 0$, this term accounts for a loss of kinetic energy, thereby allowing our energy minimizing system to eventually come to rest. It also provides a measure of inelasticity to collisions which is more typical of how we expect a synthetic material to behave. A viscous damping unit force function is given by

$$-\beta_1 \left(\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\right), \tag{3.18}$$

where β_1 is the damping coefficient and $\dot{\mathbf{x}}_i$ and $\dot{\mathbf{x}}_j$ are particle velocities. This velocity based damping force differs from damping with respect to the world reference frame, $-\gamma_i \dot{\mathbf{x}}_i$, because it is independent of rigid body motion. Instead of decreasing the momentum of a single particle, it transfers momentum between neighboring particles.



(a) Potentials



(b) Forces

Figure 3.3: Spring and Lennard-Jones compared.

Potential energies and corresponding forces shown. Spring labeled SP. Weighted Lennard-Jones labeled WLJ. Lennard-Jones is labeled LJ. The spring parameters are $r_o = 1$ and $k = 8$. The Lennard-Jones parameters are $m = 2, n = 4, e_o = 1$, and $r_o = 1$. The weighting function parameters are $r_a = r_o$, and $r_b = 2r_o$.

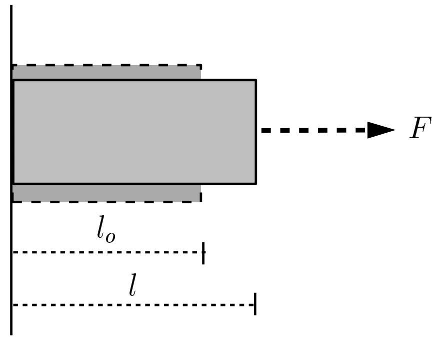


Figure 3.4: Longitudinal deformation

3.3.3 Rheology

Rheology is the science of the deformation and flow of matter. According to rheology, all materials can be modeled as falling between the two extremes of pure elastic (solid metal) and pure viscous (fluid) behavior (Houwink and de Decker, 1971). No real materials exhibit ideally elastic or ideally viscous behavior, though there are materials which come close. Some examples of elastic behavior are the steel spring and vulcanized rubber. Examples of viscous behavior include liquids such as water at room temperature, syrups, and molten glass. A few examples of mixed elastic and viscous behavior are wood, silk, PVC, and nylon.

Elasticity

A material that behaves according to ideal elastic behavior means that the material deforms under force and returns to its original form after release of the force. The resistance to deformation is described by the Young's modulus of elasticity E

$$F = E\epsilon \tag{3.19}$$

where F is the normal stress (force) and ϵ is the relative elongation

$$\epsilon = \frac{l - l_o}{l_o}$$

as shown in Figure 3.4.

The modulus of elasticity E is a physical constant only when the deformation is proportional to the stress. A linear approximation of Young's modulus can be computed using the Taylor series expansion. We compute it for small deformations near the equilibrium separation

$$F(r) = F(r_o) + (r - r_o)\frac{\mathrm{d}F(r_o)}{\mathrm{d}r} + O((r - r_o)^2).$$

Ignoring higher order terms we have

$$\begin{aligned} F(r) &= F(r_o) + \frac{\mathrm{d}F(r_o)}{\mathrm{d}r}(r - r_o) \\ &= -\frac{e_o nm}{r_o^2}(r - r_o). \end{aligned}$$

For this case, Young's modulus is

$$E = -\frac{e_o nm}{r_o}$$

and we have derived the equation for ideal linear elastic deformation, that is Hooke's law

$$F(r) = k(r - r_o),$$

where $k = E/r_o$.

Yield limit

Real materials have definite limits beyond which deformations are no longer elastic. For example, solid steel or glass wires do not yield elastically beyond 1% of their length. Nylon fibers can yield 20% of their length at room temperature. An unusual example, vulcanized rubber can yield 500% of its length (Houwink and de Decker, 1971). The limit after which materials are no longer elastic is known as the yield limit. After this limit other mechanisms take over, such as plasticity or flow behavior. Under some conditions there is no mechanism that takes over, in which case the material breaks. The pieces of such a broken object can be fitted back together to reconstruct the original shape. The breaking of ceramic coffee mugs or everyday drinking glasses are examples.

The tensile strength of a material is a measure of the maximum force value exhibited by a material in response to a stress. In the case of the Lennard-Jones function, the force increases as the particles separate until the maximum force at the separation value $r = r_t$ is reached and then it decreases until force goes to zero at infinity or the distance r_b, in the case of a weighted potential. The maximum force is when

$$\frac{d^2\phi_{\mathrm{LJ}}(r)}{dr^2} = 0,$$

which is equivalent to the condition

$$r^{n-m} = r_o^{n-m}\frac{n+1}{m+1}.$$

The magnitude of the tensile strength is the depth of the well in the force function as shown in Figure 3.3 (b). Under forces greater than the tensile strength, the bond will fracture.

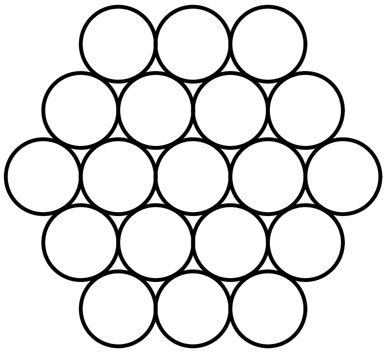


Figure 3.5: Hexagonal packing.

Plasticity

Plastic or visco-elastic behavior is the response of materials that combine both elastic and viscous responses to stress and can be characterized by the additivity of the stress. The Voight model (Jaeger, 1969) of plasticity states

$$F = E\epsilon + \eta \frac{\mathrm{d}\epsilon}{\mathrm{d}t}$$

where F is the force and η is a viscous term. Our model is similar to the Voight model with the elastic portion represented by the Lennard-Jones force and the viscous portion represented by the viscous damping unit equation (3.18).

For stresses less than the maximum force value, the particles will separate by less than the yield separation value of $r_t - r_o$, and will return to the equilibrium separation when the force is removed. For stresses greater than the tensile strength value, particle bonds may be broken and permanent deformations may occur as particles are rearranged relative to one another.

3.4 Packings

Since the Lennard-Jones function is defined in terms of the Euclidean distance from a particle, it is a *spatially symmetric* potential energy function. Given two particles i and j the set of minimum energy states (positions) for particle j relative to i is the locus of points of a sphere centered at the position of particle i. When external forces are insignificant, particles arrange into closely packed structures to minimize their total energy.

For spherically symmetrical potential energy functions in 2-D, such as the Lennard-Jones potential, the particles arrange into hexagonal orderings as shown in Figure 3.5. In 3-D the particles arrange into hexagonal ordered 2-D layers, making Lennard-Jones good for modeling volumes of material.

3.4.1 Density

The dense packing of particles lets us ask "How many particles will it take to fill a given volume?" We have found this to be a common question. The answer is also used in computing the efficiency of neighboring finding techniques given in Section 7.2. We can approximate the answer by considering the related question "How many solid equal size spheres will it take to fill a given volume?" The *volume packing factor* is defined as the ratio of the unit-sphere to the unit-enclosure (Gasson, 1983). For a given packing, this is equivalent to the ratio of the volume of a sphere to the associated 3D Voronoi region, where the Voronoi diagram is computed over the center points of the spheres. A hexagonal close packing of spheres has a volume packing factor of P_v

$$P_v = \frac{\pi}{3\sqrt{2}} \approx 0.74. \tag{3.20}$$

In comparison, a cubic packing of spheres, where each sphere is positioned at the corner of a cubic lattice, has a packing factor of $\pi/6 \approx 0.52$ which is significantly less than the hexagonal packing factor. Given the packing factor, the expected number of spheres n, in a volume V, is given by

$$n = P_v \frac{V}{V_s},$$

where $V_s = \frac{4}{3}\pi r^3$ is the volume of a sphere with radius r. For hexagonal packing this reduces to

$$n = \frac{V}{4\sqrt{2}r^3}.$$

In accordance to the Lennard-Jones force, at equilibrium two particles will be separated by the equilibrium separation r_o. Thus we approximate the volume of a particle

$$V_p = \frac{V_s}{P_v} = \frac{{r_o}^3}{\sqrt{2}}, \tag{3.21}$$

as the volume of sphere, with a radius of one half the equilibrium separation, divided by the hexagonal packing factor. When the neighborhood range includes only particles of a distance r_o this accurately represents the effective particle volume. However when the neighborhood range includes more distant particles, the particles will be packed more closely. The amount of compression depends on the strength of the attractive forces and any external pressure on the system.

3.5 External Forces

Our model of volumetric shape becomes more interesting when we put it in an environment with external forces and obstacles. In this section we describe two such external forces.

3.5.1 Gravity

We add the force of gravity, $\mathbf{f} = \mathbf{g}m_i$, where $\mathbf{g}$ is a gravitational acceleration in a given direction and m_i is the mass of particle i, to our simulated world so that we can drop objects and pour fluids.

3.5.2 Collisions

We introduce obstacles into our environment and create collision forces so that particles will not penetrate these objects. A repulsive force is defined between each particle and object surface similar to the repulsive force between particles. The repulsive force is limited to a short range so that particles are only repelled when they are very close to the surface. The force is based on inverse powers of the distance between the object and particle. As a particle and object collide, the particle will slow down due to the repulsive force and gain potential energy relative to the obstacle.

For an object k and particle i separated by a distance d_{ki}, we define the collision potential energy function as

$$e_c\left(d_{ki}\right) = \begin{cases} \alpha/d_{ki}^{\eta} + \beta d_{ki} + \gamma & \text{when } d_{ki} \leq d_o \\ 0 & \text{otherwise,} \end{cases} \tag{3.22}$$

and the resulting force as

$$f_c\left(d_{ki}\right) = \begin{cases} \frac{n\alpha}{d_{ki}^{\eta+1}} - \beta & \text{when } d_{ki} \leq d_o \\ 0 & \text{otherwise.} \end{cases} \tag{3.23}$$

The distance d_o is the distance from the obstacle surface at which the particle gains potential energy relative to the obstacle. By constraining $e_c\left(d_o\right) = 0$ and $f_c\left(d_o\right) = 0$, the potential energy and force functions are continuous for $d_{ki} > 0$. For a constant value of η, the user need only specify the distance $d_o > 0$ and the scaling factor $\alpha > 0$, and the remaining constants are uniquely determined. This works for all obstacles given there is a function for computing the shortest distance between a point and the surface.

3.6 Examples

3.6.1 Deformations

Figures 3.6 and 3.7 shows an object modeled using volume particles. The parameters of the Lennard-Jones potential are varied to make one version of the model rigid and the other flexible. In both figures the bonds between particles are strong enough so that the objects behave as solids, maintaining their structure under the influence of external forces, in this case gravity and collision forces. In this example we varied the exponents of the Lennard-Jones potential, thus increasing the maximum binding force, without increasing the dissociation energy. That is it "sharpens" the potential

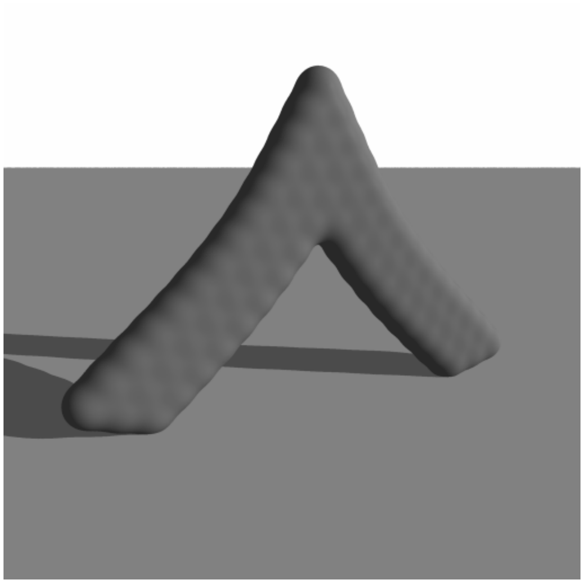


Figure 3.6: Flexible solid

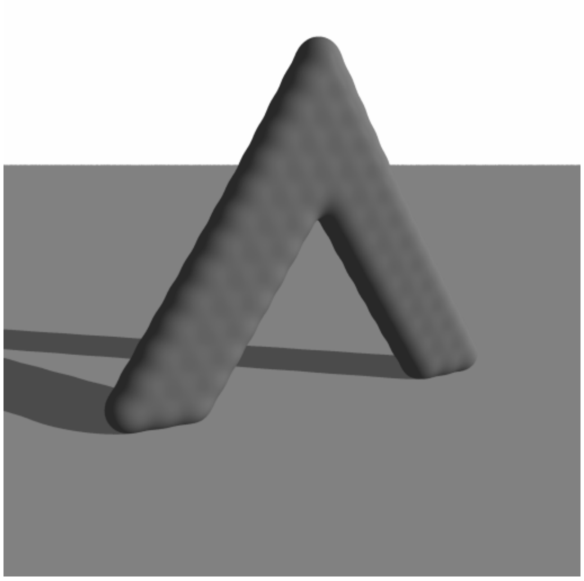


Figure 3.7: Rigid solid

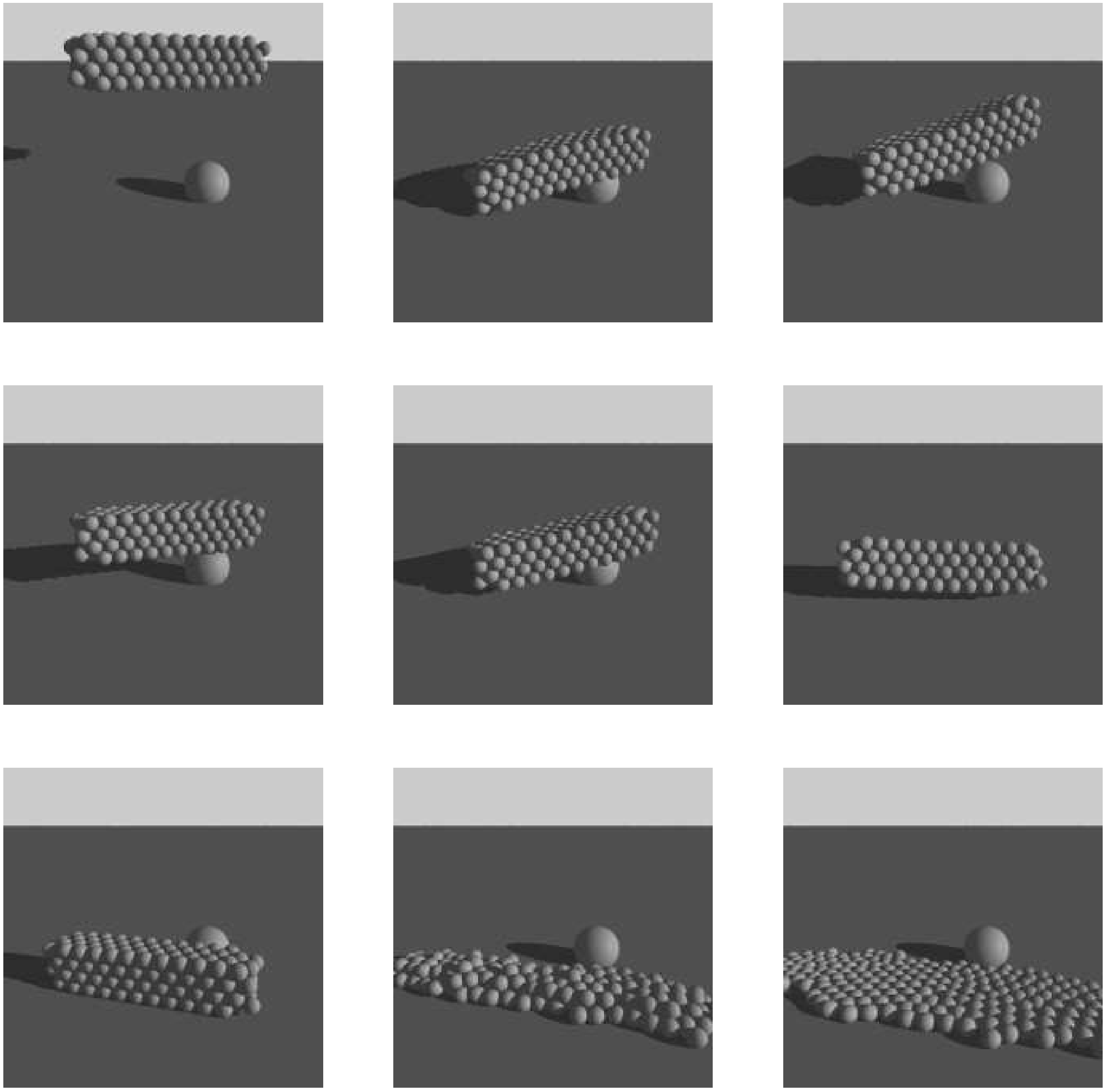


Figure 3.8: Beam colliding

A solid beam falling, colliding with obstacles. The frames were taken from an animation at the following times: $t = 0, 3, 4, 5, 9, 12, 25, 70, 83$. In the second frame the beam collides with the sphere, and floor (not shown), causing it to bounce back into the air (frames 3 and 4). Note the slight flexing of the beam. In frames 6 and 7 it rolls forward, further displaying rigid body motion. In the last two frames the magnitude of the dissociation energy is reduced (see Chapter 6) and the initial structure of the object is lost.

well without "deepening" it. This creates rigid materials, but also ones that are "brittle".

Another approach to creating rigid solids is to increase the magnitude of the dissociation energy. This also increases the maximum binding force, but creates materials that are more durable and less likely to break under sudden external forces. The bouncing beam in Figure 3.8 is an example using this approach. In this example, a solid beam falls colliding with obstacles, exhibiting rigid body motion with slight deformation. After striking a sphere and the floor plane the beam bounces back into the air. After the second collision with the sphere it rolls forward coming to rest on the ground plane. In the last few frames the dissociation energy is reduced and the solid looses its initial structure much like the varying structure found in fluids.

For Figures 3.6 and 3.7 the Lennard-Jones parameters are equilibrium separation $r_o = 1.192$, dissociation energy of $e_o = 200$, and the exponents values $(n = 4, m = 2)$ and $(n = 8, m = 6)$ for the flexible and rigid solids respectively. The remaining parameters are as follows. The particles have mass $m_i = 1$. The gravitational constant is $g = 1.5$. The collision force constants are $\beta = 25$, $d_o = 1$, and $\eta = 4$. The velocity based damping constant is $\gamma = 1.5$ and the viscous based damping constant is $\beta_1 = 0$. Each solid is composed of 69 particles. The numerical integration was performed according to the Euler method (Chapter 7) and rendered as iso-surfaces (Chapter 7).

We have begun our development of a flexible shape model which allows the user to create objects of arbitrary topology. The goal of the model is to provide a simple yet powerful method of modeling objects whose local shape and geometry change rapidly over time. An important criterion, implicit in this goal, is the ability to make large changes at the global level, such as changes in topology. It is also important for the user to be able to control the physical characteristics of the model, such as the relative amount of deformation due to a given force. We have described a particle based model suitable for representing volumes and in the next chapter we extend this model to surfaces.

The approach taken to satisfy these requirements is to represent the object as a collection of primitive elements, whose relative positions dictate the shape and geometry of the object. That is, different shapes and geometries will have different arrangements of the volume elements. Similar to the use of pixels (picture elements) to portray an unlimited range of two dimensional images, the use of simple elements can be used to describe an unlimited number of shapes.

The model, based on dynamically coupled particle systems, describes changes in geometry and the movement of element volumes, as a consequence of external forces and internal potential energies. By varying the internal potential energies we can model a variety of physical properties ranging from stiff to fluid like behavior. Timing results can be found in Section 7.3.5.

Chapter 4

Particle Surfaces

In this chapter we present a new model of elastic surfaces based on interacting particle systems. Unlike previous surface models, the new model can be used to split, join, or extend surfaces without the need for manual intervention. 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 volumes rather than surfaces. This is because traditional particles are point masses with no preferred orientation along which surfaces might form. To overcome this limitation, we introduce a 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. The particles we use have long-range attraction forces and short-range repulsion forces and follow Newtonian dynamics, like the models of volumes presented in the previous chapter.

We begin by extending the mathematics of the particle systems presented in Chapter 3 to include oriented particles. Based on concepts from differential geometry, we derive inter-particle potential functions which encourage particles to form smooth surfaces. A review of the relevant geometric concepts is provided in Appendix A.

4.1 Oriented Particle Systems

This section discusses the basic mathematics of oriented particle systems. We extend the definition of a particle system given in Chapter 3 to include the concepts of orientation, angular inertia, angular velocity, and torque. As such, we build upon our previous presentation, redefining the properties of the system, such as the definitions of a particle, the equations of motion, kinetic and potential energies, and the surface density of particles.

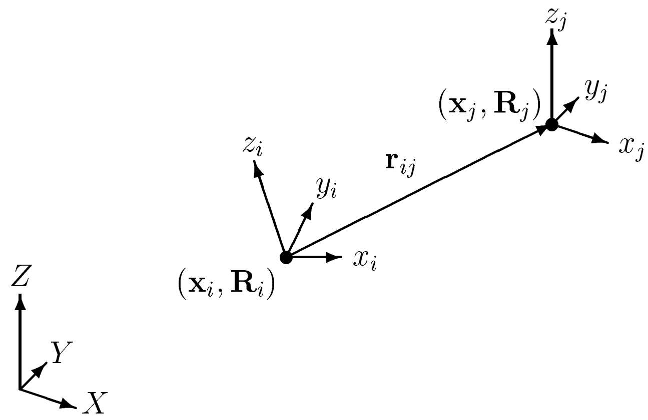


Figure 4.1: Oriented particles in global and local coordinate frames.

The global inter-particle distance $\mathbf{r}_{ij}$ is computed from the global coordinates $\mathbf{x}_i$ and $\mathbf{x}_j$ of particles i and j. The local distance $\mathbf{d}_{ij}$ is computed from $\mathbf{r}_{ij}$ and the rotation matrix $\mathbf{R}_i$.

4.1.1 The Oriented Particle

An oriented particle, like the previously discussed volume based particle, has a position and mass. In addition, each oriented particle has an orientation and inertia tensor. The orientation defines both a normal vector (z in Figure 4.1) 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 $(\mathbf{x}_i, \mathbf{R}_i)$, where $\mathbf{x}_i$ 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 $\mathbf{n}_i$. While we define rotation as a matrix for conversion between local and global coordinates and vice versa, we use unit quaternions to store the rotation in practice. The unit quaternion

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

represents a rotation of θ about the unit normal axis $\mathbf{a}$. 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). Our use of rotations is discussed in more detail in Appendix B.

The inertia tensor $\mathbf{I}$ relates the angular momentum to the angular velocity by a linear transformation. The angular velocity of a particle describes the rotation of a particle about the particle's local origin. The inertia tensor is defined about the particle's local origin with respect to the world coordinate axes. Since we are interested in the property of angular momentum but not a particular rigid body we choose a simple inertia tensor, one that is a constant scaling of the identity matrix. This is valid for all positions and orientations of a given particle. We discuss the inertia tensor in more detail in Appendix B.

4.1.2 Equations of Motion

A system of oriented particles is governed by the set of ordinary differential equations of motion: equation (3.1) rewritten here

$$m_i \ddot{\mathbf{x}}_i + \gamma_i \dot{\mathbf{x}}_i + \mathbf{f}_i^{\mathtt{int}} = \mathbf{f}_i^{\mathtt{ext}} \qquad i = 1, \ldots, N$$

and for our choice of inertia tensor, the associated equation for angular motion

$$\mathbf{I}_i \ddot{\mathbf{q}}_i + \xi_i \dot{\mathbf{q}}_i + \boldsymbol{\tau}_i^{int} = \boldsymbol{\tau}_i^{ext} \qquad i = 1, \ldots, N \tag{4.1}$$

where i is the particle index, N is the number of particles in the system, $\ddot{\mathbf{x}}_i$ is the acceleration of a particle, m_i is its mass, $\dot{\mathbf{x}}_i$ is its velocity, γ_i is a translational damping coefficient that controls the rate of dissipation of the particle's translational kinetic energy, $\mathbf{f}_i^{\mathtt{int}}$ is the sum of inter-particle forces, $\mathbf{f}_i^{\mathtt{ext}}$ is the sum of external forces, $\mathbf{q}_i$ is the orientation of particle i in three-space, $\ddot{\mathbf{q}}_i$ is the angular acceleration, $\mathbf{I}_i$ is the angular inertia tensor (B.1), $\dot{\mathbf{q}}_i$ is the angular velocity, ξ_i is an angular damping coefficient that controls the rate of dissipation of the particle's rotational kinetic energy, $\boldsymbol{\tau}_i^{int}$ is the sum of inter-particle torques and $\boldsymbol{\tau}_i^{ext}$ is the sum of external torques.

The inter-particle internal force term (3.2) is redefined

$$\mathbf{f}_i^{\mathtt{int}}(\mathbf{x}_1, \mathbf{q}_1, \mathbf{x}_2, \mathbf{q}_2, \ldots, \mathbf{x}_N, \mathbf{q}_N) \tag{4.2}$$

as a function of position *and orientation*. The external force term (3.3) is redefined as

$$\mathbf{f}_i^{\mathtt{ext}}(\mathbf{x}_i, \mathbf{q}_i, \mathbf{S}) \tag{4.3}$$

a function of particle position, orientation, and and the set $\mathbf{S}$ of external state variables such as gravity and collision objects. The inter-particle torque terms are defined in a similar manner

$$\boldsymbol{\tau}_i^{\mathtt{int}}(\mathbf{x}_1, \mathbf{q}_1, \mathbf{x}_2, \mathbf{q}_2, \ldots, \ldots, \mathbf{x}_N, \mathbf{q}_N). \tag{4.4}$$

External torques are functions of position, orientation, and external state variables

$$\boldsymbol{\tau}_i^{\mathtt{ext}}(\mathbf{x}_i, \mathbf{q}_i, \mathbf{S}). \tag{4.5}$$

4.1.3 Potential Energy

Similar to the use of potential energies for un-oriented particle systems, potential energies provide a convenient description of forces and torques between particles. This formulation also guarantees that the system will not diverge. For a potential function ϕ, the force exerted on particle i with position $\mathbf{x}_i$

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i} \phi$$

is due to the gradient of the potential energy ϕ with respect to the change in position. The *torque* exerted on particle i with orientation $\mathbf{q}_i$

$$\boldsymbol{\tau}_i = -\nabla_{\theta_i} \phi$$

is due to the gradient of the potential energy with respect to the incremental change in orientation θ_i.

For oriented particle systems we also adopt the pairwise additivity assumption which states that the total potential energy of a particle (3.4) is the sum of the pairwise potential energies between that particle and every other particle. Equation (3.5) describes the total inter-particle forces acting on a particle i due to the inter-particle potential energy functions and

$$\boldsymbol{\tau}_i = -\nabla_{\theta_i}\phi_i = -\sum_{j\neq i}^{N} \nabla_{\theta_i}\phi_{ij} \tag{4.6}$$

describes the total inter-particle torques acting on a particle due to the inter-particle potential energy functions. The equation for the total potential energy of the system (3.6) is the same for oriented particles as for un-oriented particles.

4.1.4 Kinetic Energy

The kinetic energy of an oriented particle is a combination of translational and rotational kinetic energies. The translational kinetic energy K_i of a single oriented particle i is given by equation (3.7). In order to compute the rotational kinetic energy separate from the translational kinetic energy, it is, in general, necessary to choose a inertial coordinate system whose origin is the centroid of the object (Marion, 1970). For a single particle, the rotational kinetic energy of a particle is given by

$$K_{\texttt{rot}} = \frac{1}{2}\sum_{j,k} \mathbf{I}_{jk}\omega_j\omega_k$$

where the subscripts j and k refer to individual elements of the interia tensor $\mathbf{I}$. Our choice of inertia tensor (B.1) allows us to simplify the rotational kinetic energy to be

$$K_{\texttt{rot}} = \frac{1}{2}\left(I_{xx}{\boldsymbol{\omega}_x}^2 + I_{yy}{\boldsymbol{\omega}_y}^2 + I_{zz}{\boldsymbol{\omega}_z}^2\right) \tag{4.7}$$

for the principal moments of inertia I_{xx}, I_{yy}, and I_{zz}. For additional details see Appendix B.2.

We write the total kinetic energy of the system as the sum of the individual particle translational (3.7) and rotational kinetic energies (4.7),

$$E_K = \sum_{i}^{N} \left(K_i + K_{\texttt{rot}i}\right). \tag{4.8}$$

This redefines the system kinetic energy equation (3.8).

4.1.5 System Energy

The total energy of the system, E_S, is simply a summation of the individual particle kinetic and potential energies

$$E_S = E_K + E_P, \tag{4.9}$$

where E_P is given by equation (3.6) and E_K is given by equation (4.8).

4.1.6 Angular Damping

The second term of (4.1) is an angular velocity based damping force, $-\xi_i \dot{\mathbf{q}}_i$. When $\xi_i > 0$ this term accounts for a loss of rotational kinetic energy. This is analogous to the translational damping force $-\gamma_i \dot{\mathbf{x}}_i$. Analogous to the ideal viscous damping unit between neighboring particles is the angular viscous damping unit function

$$-\beta_3 \left(\dot{\mathbf{q}}_i - \dot{\mathbf{q}}_j \right), \tag{4.10}$$

where β_3 is the damping coefficient and $\dot{\mathbf{q}}_i$ and $\dot{\mathbf{q}}_j$ are the angular velocities of particles i and j.

4.1.7 Dynamic Coupling

To compute all inter-particle forces and torques without restrictive assumptions on the potential energies requires N^2 force computations. This is similar to the unoriented particle case. Assuming particles are uniformly distributed as a sampling of a surface, each particle will have at on average a constant number of neighbors, and by ignoring distant particles the inter-particle force calculations for the system are reduced to $O(N)$ computations. Thus we can rewrite the force and energy equations in terms of the set of nearest neighbors. This is a straightforward procedure and since we have already discussed this concept in Chapter 3, we omit deriving the equations for the oriented case.

4.1.8 Weighting Function

In Section 3.2.1 we discussed a weighting function (3.12) which monotonically decreases to zero at the distance of the particle neighborhood range. We have experimented with two other weighting functions. The first one

$$w_2(r) = e^{-r^2/2\sigma_r^2}$$

is based on the Gaussian distribution with $\sigma_r = 1.0$. The second weighting function

$$w_3(x, y, z) = K e^{\left(-\frac{x^2+y^2}{2a^2} - \frac{z^2}{2b^2}\right)} \qquad b \le a \tag{4.11}$$

generalizes this to favor interactions between particles that lie close to their respective tangent planes. We accomplish this by writing the function in terms of the particle's local coordinates, e.g., by replacing the inter-particle distance $\mathbf{r}_{ij}$ by

$$\mathbf{d}_{ij} = \mathbf{R}_i^{-1} \mathbf{r}_{ij} = \mathbf{R}_i^{-1} (\mathbf{x}_j - \mathbf{x}_i), \tag{4.12}$$

where $\mathbf{d}_{ij} = [x, y, z]^T$, and $\mathbf{R}_i$ is the particle orientation. That is $\mathbf{d}_{ij}$ is the position of particle j in particle i's local coordinate frame.

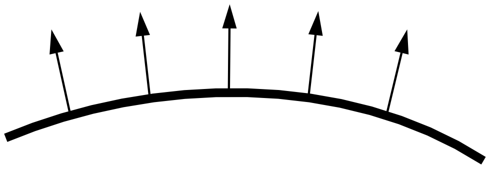


Figure 4.2: Cross sectional view of a of surface

4.1.9 Density

Assuming that a set of particles present a uniform sampling of a surface we can compute the density of particles over the surface. Similar to the volume case we ask the question: "How many particles will it take to fill a given area?" We can approximate the answer by considering the related question: "How many solid equal size circles will it take to fill a given area?" The *area packing factor* is defined as the ratio of the unit-circle to the unit-enclosure (Gasson, 1983). For a given packing, this is equivalent to the ratio of the area of a circle to the associated 2D Voronoi region, where the Voronoi diagram is computed over the center points of the circles. A hexagonal close packing of circles has an area packing factor P_A of

$$P_A = \frac{\pi}{2\sqrt{3}} \approx 0.91.$$

The expected number of circles n in a given area A, is given by

$$n = P_A \frac{A}{A_c} = \frac{A}{4\sqrt{3}r^2},$$

where A_c is the area of the circle with radius r. We can approximate the area of a surface particle A_p

$$A_p = \frac{A_c}{P_A} = \frac{\sqrt{3}}{2}{r_o}^2$$

as the area of a circle, with a radius of one half the Lennard-Jones equilibrium separation r_o, divided by the hexagonal packing factor.

4.2 Surface Potentials

To encourage oriented particles to group themselves into surface-like arrangements, we devise a collection of new inter-particle potential functions. These potential functions can be derived from the deformation energies of local triangular patches using finite element analysis, or from the differential geometry of surfaces. We begin with an intuitive explanation based on analogies with physical surfaces. We follow with an analysis correlating the potential functions to the geometric measures of differential geometry. We defer the details of the finite element analysis to Appendix E.

We derive our potentials by considering an infinitesimally small section of a surface as shown in Figure 4.2. Over a small section one can notice that adjacent points on

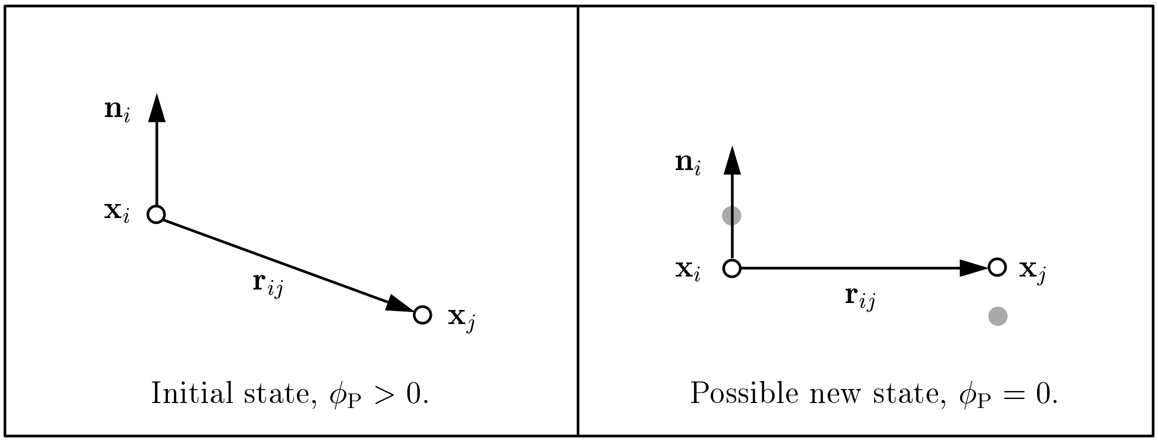


Figure 4.3: Interaction due to co-planarity potential: $\phi_P = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2$

The original particle positions are drawn in black on left and in gray on right.

the surface have normals that are close to parallel and the points lie near the tangent planes of adjacent points. One can also note that for circular arcs the normals diverge with equal angles. We use these observations to define potential energy functions between adjacent particles which encourage such configurations. We define three potentials: a co-planarity potential to encourage particles to lie in neighboring particle tangent planes, a co-normality potential to encourage particle normals to align, and for surfaces with constant curvature, a co-circularity potential to encourage particle normals to diverge with equal angle.

4.2.1 Co-planarity

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

$$\phi_P(\mathbf{n}_i, \mathbf{r}_{ij}) = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \tag{4.13}$$

The energy is proportional to the scalar product between the surface normal and the vector to the neighboring particle. Recall that two nonzero vectors are perpendicular if and only if their scalar product is zero.

We illustrate this in Figure 4.3. On the left side is a non-zero potential energy state. On the right side the particles have obtained a zero potential energy state, by moving in directions parallel to the normal vector $\mathbf{n}_i$. The forces acting on the two particles are given by the gradient of the potential energy with respect to $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$\begin{aligned} \mathbf{f}_i &= -\nabla_{\mathbf{x}_i}\phi_P = 2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i \\ \mathbf{f}_j &= -\nabla_{\mathbf{x}_j}\phi_P = -2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i \end{aligned}$$

For the example given, the product $\mathbf{n}_i \cdot \mathbf{r}_{ij}$ is a negative scalar value. Thus particle i moves in the direction of $-\mathbf{n}_i$ and particle j moves in the direction of $+\mathbf{n}_i$. The torques

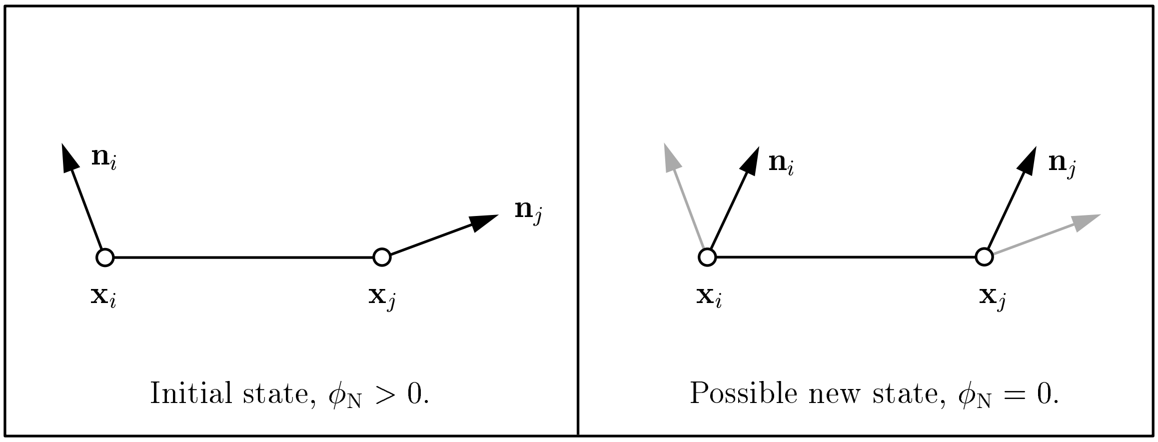


Figure 4.4: Interaction due to co-normality potential: $\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2$

The original particle normals are drawn in black on left and in gray on right.

are given by the gradient of the potential energy with respect to the orientation:

$$\begin{aligned}\boldsymbol{\tau}_i &= -\nabla_{\theta_i}\phi_{\mathrm{P}} = -2(\mathbf{n}_i \cdot \mathbf{r}_{ij})(\mathbf{n}_i \times \mathbf{r}_{ij}) = \mathbf{r}_{ij} \times \mathbf{f}_i \\ \boldsymbol{\tau}_j &= -\nabla_{\theta_j}\phi_{\mathrm{P}} = 0.\end{aligned}$$

For the example given, the torque vector on particle i would be into the page and result in rotating the particle clockwise until $\mathbf{n}_i$ and $\mathbf{r}_{ij}$ form a right angle. There is no torque on particle j because the potential is independent of j's normal. However when we evaluate the potential $\phi_{\mathrm{P}}(\mathbf{n}_j, \mathbf{r}_{ji})$ then the situation reverses, $\boldsymbol{\tau}_i$ will be zero and $\boldsymbol{\tau}_j$ will be non-zero.

4.2.2 Co-normality

The co-planarity potential does not control the "twist" in the surface between two particles. To limit this, we introduce a *co-normality* potential

$$\phi_{\mathrm{N}}(\mathbf{n}_i, \mathbf{n}_j) = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \tag{4.14}$$

which attempts to line up neighboring normals, much like interacting magnetic dipoles.

We illustrate this in Figure 4.4. On the left side the normals are not parallel resulting in a non-zero potential energy state. On the right side the particles have obtained a zero potential energy energy state by rotating until their normals align. The torques acting on the two particles is given by the gradient of the potential energy with respect to orientation:

$$\boldsymbol{\tau}_i = -\nabla_{\theta_i}\phi_{\mathrm{N}} = 2(\mathbf{n}_j \times \mathbf{n}_i) \tag{4.15}$$

$$\boldsymbol{\tau}_j = -\nabla_{\theta_j}\phi_{\mathrm{N}} = -2(\mathbf{n}_j \times \mathbf{n}_i) \tag{4.16}$$

The torque is about an axis orthogonal to $\mathbf{n}_i$ and $\mathbf{n}_j$, applied equally to both particles. For the example given it induces a clockwise rotation on particle i and a counter-clockwise rotation on particle j. This potential is independent of particle positions and

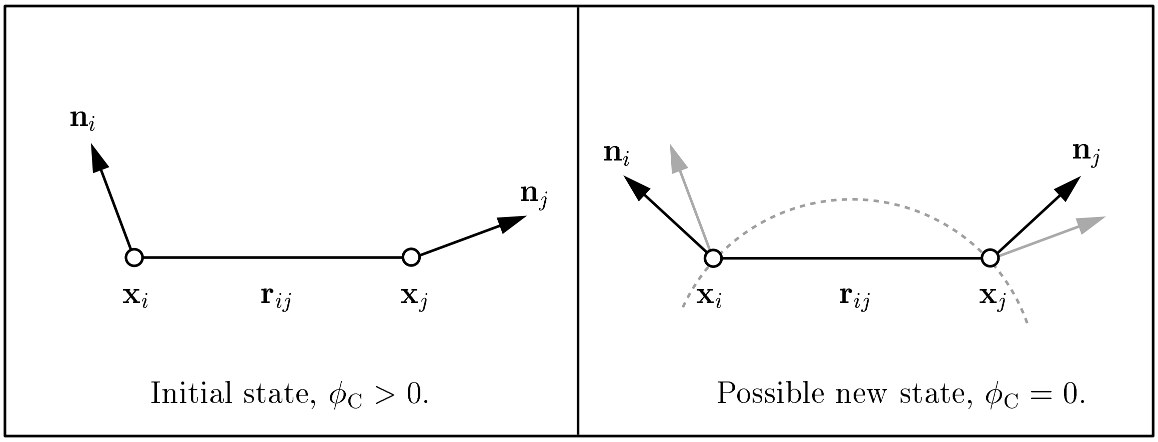


Figure 4.5: Interaction due to co-circularity potential: $\phi_C = ((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij})^2$

The original particle normals are drawn in black on left and in gray on right. The dashed line shows an arc of constant curvature matching the new normals and passing through the particle positions.

---

thus the resulting forces are zero. By itself, the co-normality potential is not sufficient to form surfaces as the tangent planes may not align, as seen in this example. However in combination with the co-planarity potential, energy minimization will encourage a set of uniformly spaced particles to arrange into a continuous surface.

4.2.3 Co-circularity

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

$$\phi_C(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij}) = ((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij})^2 \tag{4.17}$$

which is zero when normals are anti-symmetrical with respect to the vector joining the two particles. This is the natural configuration for surface normals on a sphere. That is the energy is zero under the condition that the angle between particle i's normal and the separation vector $\mathbf{r}_{ij}$ is equal to the angle between particle j's normal and the separation vector, or $\mathbf{n}_i \cdot \mathbf{r}_{ij} = \mathbf{r}_{ji} \cdot \mathbf{n}_j$. Since $\mathbf{r}_{ji} = -\mathbf{r}_{ij}$ and the scalar product is distributive we can rewrite the condition $\mathbf{n}_i \cdot \mathbf{r}_{ij} = \mathbf{r}_{ji} \cdot \mathbf{n}_j$ to be $(\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} = 0$.

We illustrate the co-circularity potential in Figure 4.5. On the left side the angles between normals and the separation vector are unequal. On the right side the particles have obtained a zero potential energy state by rotating until their normals are anti-symmetric. The torques acting on the two particles is given by the gradient of the potential energy with respect to orientation

$$\boldsymbol{\tau}_i = -\nabla_{\theta_i}\phi_C = \alpha(\mathbf{n}_i \times \mathbf{r}_{ij}) \tag{4.18}$$
$$\boldsymbol{\tau}_j = -\nabla_{\theta_j}\phi_C = \alpha(\mathbf{n}_j \times \mathbf{r}_{ij}), \tag{4.19}$$

where $\alpha = 2((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij})$ is a scalar. For each particle the torque is about an axis orthogonal to the particle normal and the separation vector. For the example given it

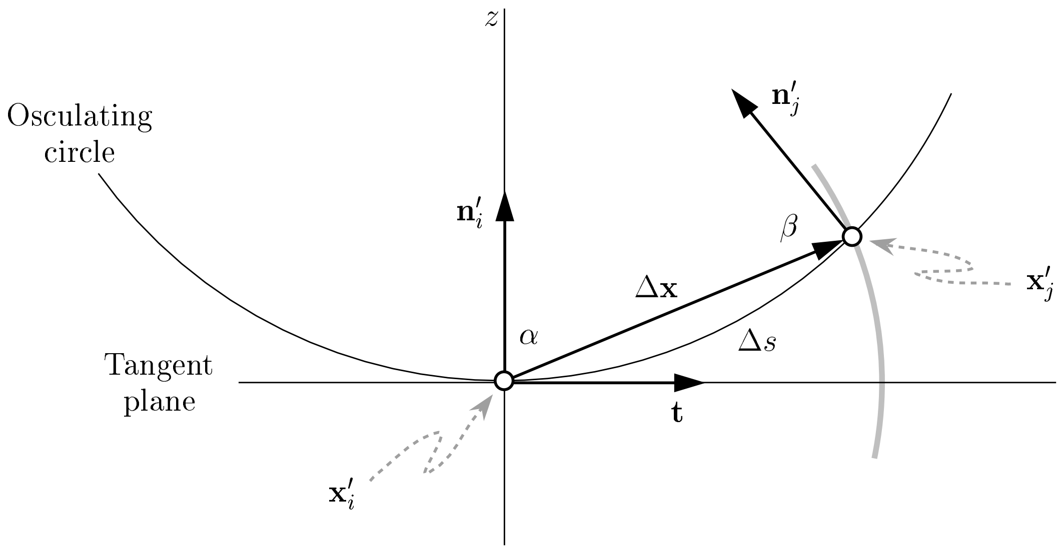


Figure 4.6: The osculating circle in the local coordinate frame of particle i

induces a counter-clockwise rotation on particle i and a clockwise rotation on particle j. The forces resulting from the change in potential are

$$\begin{aligned}\mathbf{f}_i &= -\nabla_{\mathbf{x}_i}\phi_{\mathrm{C}} = 2\left((\mathbf{n}_i + \mathbf{n}_j)\cdot\mathbf{r}_{ij}\right)(\mathbf{n}_i + \mathbf{n}_j)\\ \mathbf{f}_j &= -\nabla_{\mathbf{x}_j}\phi_{\mathrm{C}} = -2\left((\mathbf{n}_i + \mathbf{n}_j)\cdot\mathbf{r}_{ij}\right)(\mathbf{n}_i + \mathbf{n}_j) = -\mathbf{f}_{\mathrm{C}_i}.\end{aligned}$$

For particle i, the term $((\mathbf{n}_i + \mathbf{n}_j)\cdot\mathbf{r}_{ij})$ is a scalar value that is zero when the particles are anti-symmetric. The force on particle i is in the direction $(\mathbf{n}_i+\mathbf{n}_j)$ and for particle j in the opposite direction. This in effect rotates the line segment $\mathbf{r}_{ij}$ in the plane defined by $\mathbf{r}_{ij}$ and $(\mathbf{n}_i + \mathbf{n}_j)$. For the example given it rotates $\mathbf{r}_{ij}$ clockwise.

4.3 Geometric Interpretation in Local Coordinates

Another viewpoint relating the potentials to the local geometry of the surface may be found by considering the potentials in the context of a particle's local reference frame. This transformation results in simplified equations for the potentials.

We begin by writing the states of particles i and j in the local reference frame of particle i, where $\mathbf{R}_i$ is the particle orientation, and where we denote local coordinates by a prime $'$:

$$\begin{aligned}\mathbf{x}'_i &= \mathbf{R}_i^{-1}(\mathbf{x}_i - \mathbf{x}_i) = [0,0,0]^T && (4.20)\\ \mathbf{x}'_j &= \mathbf{R}_i^{-1}(\mathbf{x}_j - \mathbf{x}_i) = [x_j, y_j, z_j]^T && (4.21)\\ \mathbf{n}'_i &= \mathbf{R}_i^{-1}\mathbf{n}_i = [0,0,1]^T && (4.22)\\ \mathbf{n}'_j &= \mathbf{R}_i^{-1}\mathbf{n}_j = [n_x, n_y, n_z]^T && (4.23)\\ \mathbf{r}'_{ij} &= \Delta\mathbf{x} = \mathbf{x}'_j - \mathbf{x}'_i = [x_j, y_j, z_j]^T && (4.24)\end{aligned}$$

Figure 4.6 illustrates two particles in local coordinates with $\mathbf{x}'_i$ and $\mathbf{x}'_j$ lying in the osculating plane containing $\mathbf{x}'_i$, $\mathbf{x}'_j$, and $\mathbf{n}'_i$. Note that $\mathbf{x}'_i$ is at the origin, the normal $\mathbf{n}'_i$ is in the direction of the z axis and the tangent vector to the curve is orthogonal to the normal. The osculating circle tangent at $\mathbf{x}'_i$ is also shown. In this example the normal $\mathbf{n}'_j$ lies in the osculating plane, although in other cases it may not. The length of the curve segment from $\mathbf{x}'_i$ to $\mathbf{x}'_j$ is labeled as Δs. The chord of the segment is the difference of positions and is labeled $\Delta \mathbf{x}$. If we assume the particles maintain a constant separation distance, then as the curvature varies the point $\mathbf{x}'_j$ moves along the gray arc. As the curvature goes to zero, $\mathbf{x}'_j$ moves down to the tangent axis, and Δs tends to the chord length $\|\Delta \mathbf{x}\|$.

In local coordinates the co-planarity potential reduces to

$$\phi_{\mathrm{P}} = (\mathbf{n}'_i \cdot \mathbf{r}'_{ij})^2 = (z_j)^2. \tag{4.25}$$

Minimizing the potential, can be interpreted as encouraging particle j to reduce z_j or, in other words, to move to particle i's tangent plane. Since we are in i's local coordinate this can also be accomplished by rotating and/or moving particle i in the world frame. These three possibilities correspond to the three effects (f_i, f_j, and t_i) that were derived in Section 4.2.1.

The co-normality potential reduces to

$$\phi_{\mathrm{N}} = \|\mathbf{n}'_i - \mathbf{n}'_j\|^2 = 2(1 - n_z). \tag{4.26}$$

By noting $\|\mathbf{n}'_j\| = 1$, the minimum is clearly when $n_z = 1$, or in other words when particle j's normal aligns with the z axis.

The co-circularity potential reduces to

$$\phi_{\mathrm{C}} = \Big((\mathbf{n}'_i + \mathbf{n}'_j) \cdot \mathbf{r}'_{ij}\Big)^2 = \Big((\mathbf{n}'_j \cdot \mathbf{r}'_{ij}) + z_j\Big)^2. \tag{4.27}$$

For this case the interpretation is less apparent, yet with a little algebra it becomes clear. Let α be the angle between $\mathbf{n}'_i$ and $\mathbf{r}'_{ij}$ and let β be the angle between $\mathbf{n}'_j$ and $\mathbf{r}'_{ji}$; where $\mathbf{r}'_{ij} = -\mathbf{r}'_{ji}$. The angle α is given by $\mathbf{n}'_i \cdot \mathbf{r}'_{ij} = \|\mathbf{r}'_{ij}\| \cos\alpha = z_j$. The angle β is given by $\mathbf{n}'_j \cdot \mathbf{r}'_{ji} = \|\mathbf{r}'_{ji}\| \cos\beta = -(\mathbf{n}'_j \cdot \mathbf{r}'_{ij})$. The potential (4.27) is obviously at its minimum (zero) when $z_j = -(\mathbf{n}'_j \cdot \mathbf{r}_{ij})$, or in other words when the angles α and β are equal.

4.4 Curvature

In this section we present a definition of discrete curvature for our oriented particle system. We then show that minimizing the potential energy functions is equivalent to minimizing the squared curvature defined for space curves between pairs of particles. We will also prove that we minimize the magnitude of the sum of squared curvature and torsion measures of space curves defined by pairs of particles. These space curves can be thought of as approximating the normal sections embedded in a surface interpolating the particles. Furthermore as the particle separation goes to zero the space

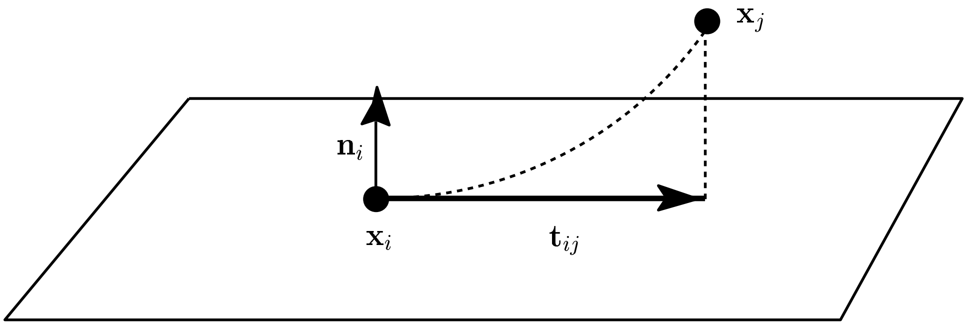


Figure 4.7: Discrete curvature tangent vector

curves between particles become osculating circles of normal sections embedded in the surface. In the limit, we prove that minimizing our potential functions is equivalent to minimizing the squared normal curvature over the surface.

We begin with a review of the basic definitions of normal sections, normal curvature, curvature, and torsion. Given points on a surface and direction tangent to the surface, there exists a unique plane that is defined by the tangent vector and surface normal that includes the given point. The intersection of the plane and the surface defines a space curve embedded in the surface. This curve is called a *normal section*. The curvature of a normal section is the *normal curvature* k_n and is defined as being the curvature of the embedded space curve. The curvature of a space curve is defined as the magnitude of the second derivative of position with respect to arc length, $\kappa = ||\dot{\mathbf{t}}|| = ||\ddot{\mathbf{x}}||$. The torsion of a space curve is defined as the magnitude of the derivative of the bi-normal with respect to arc length, $\tau = ||\dot{\mathbf{b}}||$. We defer a more detailed discussion of differential geometry to Appendix A.

4.4.1 Discrete Curvature

We now define a discrete curvature measure κ_d for our particle system. The discrete curvature is defined for a given position and direction. The position is the position $\mathbf{x}_i$ of a given particle and the direction is in a tangent direction $\mathbf{t}_{ij}$ of an adjacent particle $\mathbf{x}_j$. The tangent direction is the vector formed by $\mathbf{x}_i$ and the projection of $\mathbf{x}_j$ onto particle i's tangent plane, as shown in Figure 4.7.

Without loss of generality, we will work in the the local coordinates of particle i, where $'$ indicates local coordinates as given in Section 4.3. The position of $\mathbf{x}'_i$ is $[0,0,0]^T$, the normal $\mathbf{n}'_i$ is $[0,0,1]^T$, and the position of $\mathbf{x}'_j$ is $[x_j, y_j, z_j]^T$. The unit tangent vector $\hat{\mathbf{t}}_{ij}$ is given by $[x_j/t, y_j/t, 0]^T$ where $t = (x_j^2 + y_j^2)^{1/2}$.

We now determine the circle with normal matching $\mathbf{n}'_i$ at $\mathbf{x}'_i$ and passing through the points $\mathbf{x}'_i$ and $\mathbf{x}'_j$. We do this in a 2D plane where the x axis of this plane aligns with the tangent vector $\hat{\mathbf{t}}_{ij}$ and the y axis aligns with the particle normal $\mathbf{n}'_i$. We pick the three points

$$(x_1, y_1) \quad = \quad (0,0)$$

$$\begin{aligned}(x_2, y_2) &= (t, z_j) \\ (x_3, x_3) &= (-t, z_j)\end{aligned}$$

where (x_3, y_3) is the reflection of $\mathbf{x}'_j$ about particle i's normal. The equation

$$x^2 + y^2 + 2dx + 2ey + f = 0 \tag{4.28}$$

defines a circle of radius $(d^2+e^2-f)^{1/2}$ in the xy plane, when $d^2+e^2 > f$ (Zwillinger, 1995). Three non-collinear points determine a unique circle. For the three such points (x_1, y_1), (x_2, y_2), and (x_3, y_3) the equation

$$\begin{vmatrix} (x^2+y^2) & x & y & 1 \\ (x_1^2+y_1^2) & x_1 & y_1 & 1 \\ (x_2^2+y_2^2) & x_2 & y_2 & 1 \\ (x_3^2+y_3^2) & x_3 & y_3 & 1 \end{vmatrix} = 0 \tag{4.29}$$

defines the unique circle passing through the points. Expanding the determinant gives

$$2z_j(x^2+y^2) - 2(t^2+z_j^2)y = 0.$$

as our equation of the circle. Writing in the form of (4.28)

$$\begin{aligned} d &= 0 \\ e &= -\frac{(t^2+z_j^2)}{2z_j} \\ f &= 0 \end{aligned}$$

and thus the radius R is

$$R = \frac{x_j^2+y_j^2+z_j^2}{2z_j}.$$

We define the discrete curvature

$$\kappa_d = \frac{2z_j}{x_j^2+y_j^2+z_j^2}, \tag{4.30}$$

as the curvature of the unique circle passing through $\mathbf{x}'_i$ and $\mathbf{x}'_j$, such that at point $\mathbf{x}'_i$ the tangent and normal of the circle match $\hat{\mathbf{t}}_{ij}$ and $\mathbf{n}'_i$. In the global reference frame the discrete curvature is given as

$$\kappa_d = \frac{2}{\|\mathbf{r}_{ij}\|^2}(\mathbf{n}_i \cdot \mathbf{r}_{ij}). \tag{4.31}$$

4.4.2 Minimizing Discrete Curvature

We show that minimizing the co-planarity potential for a pair of particles, with particle spacing constrained by the Lennard-Jones potential, minimizes the squared discrete curvature measure of that pair.

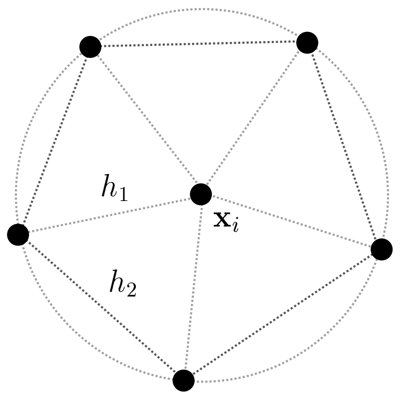


Figure 4.8: Inter-particle spacings h_1 and h_2.

In local coordinates, the co-planarity potential (4.25) is

$$\phi_{\mathrm{P}} = z_j^2$$

and the discrete curvature (4.30) squared is

$$\kappa_d{}^2 = \frac{4z_j^2}{\|\mathbf{r}'_{ij}\|^4}.$$

Since infinite energies are excluded from our simulations, we can assume that

$$0 < r_{\min} \leq \|\mathbf{r}_{ij}\|.$$

If not, the Lennard-Jones potential energy would be infinite. For completeness we mention that $\|\mathbf{r}_{ij}\| = \|\mathbf{r}'_{ij}\|$. The minimum bound $r_{\min}$ on the separation distance bounds the discrete curvature

$$\left(r_{\min}^4 \leq \|\mathbf{r}'_{ij}\|^4.\right) \;\Rightarrow\; \left(\kappa_d^2 \leq \frac{4z_j^2}{r_{\min}^4}\right).$$

Minimizing the co-planarity potential minimizes the squared discrete curvature between particle pairs

$$\begin{aligned} \phi_{\mathrm{P}} \to \min \;\; &\Rightarrow \;\; z_j^2 \to 0 \\ &\Rightarrow \;\; \kappa_d^2 \to 0. \end{aligned}$$

4.4.3 Normal Curvature

We show that our discrete curvature measure is equivalent to the normal curvature of a surface in the limit of infinitesimal particle separation. Let us assume we have a particle i surrounded by a set of symmetrically spaced particles $\mathcal{N} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N\}$ as shown in Figure 4.8. Let h_1 be the distance between each particle pair $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{N}$. Let h_2 be the distance between adjacent neighboring particles.

As the separating length h_2 approaches zero, N goes ∞, and the particles surrounding $\mathbf{x}_i$ now form a continuous circle. Thus our discrete curvature measure κ_d is defined for all tangent vectors in particle i's tangent plane.

As the separating length h_1 goes to zero, we define the curvature measure

$$\kappa_d^\star = \lim_{h_1 \to 0} \kappa_d.$$

It should be clear that the circle defined between our particle pairs (as in Figure 4.6) becomes the osculating circle of some space curve passing through the two particles. Thus the discrete curvature $\kappa_d^\star$ becomes the curvature κ of the space curve.

We now relate κ to the normal curvature measure. For any smooth surface, a point on the surface, and a tangent direction, there exists a plane that is normal to the surface and which also includes the point and tangent vector. The space curve lying in that plane and embedded in the surface is the normal section with a curvature called the normal curvature κ_n. Since the circle we define lies in the plane normal to particle i, in the limit as h_1 tends to zero, this circle becomes the normal section, and our discrete curvature becomes the normal curvature, that is $\kappa_d^\star = \kappa = \kappa_n$.

4.4.4 Minimizing Torsion

This proof shows that minimizing the co-normality potential, with particle spacing constrained by the Lennard-Jones potential, minimizes the sum of the squared curvature and squared torsion measures of a space curve. We consider a space curve passing through the points $\mathbf{x}_i$ and $\mathbf{x}_j$, such that the unit normal vectors of the curve at $\mathbf{x}_i$ and $\mathbf{x}_j$ match the particle normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ respectively.

An equation for the rate of change of the normal per unit arc length of a space curve is given by the Frenet-Serret formulas (A.2)

$$\dot{\mathbf{m}} = -\kappa\mathbf{t} + \tau\mathbf{b}. \tag{4.32}$$

For a discrete system, the change in normal is

$$\dot{\mathbf{m}} = \frac{\mathbf{n}_j - \mathbf{n}_i}{\Delta s}, \tag{4.33}$$

where Δs is the unit arc length. We combine (4.32) and (4.33) to get

$$-\kappa\mathbf{t} + \tau\mathbf{b} = \frac{\mathbf{n}_j - \mathbf{n}_i}{\Delta s}.$$

We now take the scalar product of both sides

$$\kappa^2(\mathbf{t}\cdot\mathbf{t}) - 2\kappa\tau(\mathbf{t}\cdot\mathbf{b}) + \tau^2(\mathbf{b}\cdot\mathbf{b}) = \frac{\|\mathbf{n}_j - \mathbf{n}_i\|^2}{\Delta s^2}. \tag{4.34}$$

The tangent and bi-normal are orthogonal and unit vectors so the following hold

$$\begin{aligned} \mathbf{t}\cdot\mathbf{t} &= 1 \\ \mathbf{b}\cdot\mathbf{b} &= 1 \\ \mathbf{t}\cdot\mathbf{b} &= 0. \end{aligned}$$

Equation (4.34) thus reduces to

$$\kappa^2 + \tau^2 = \frac{\|\mathbf{n}_j - \mathbf{n}_i\|^2}{\Delta s^2}. \tag{4.35}$$

We can now rewrite (4.35) in terms of the co-normality potential (4.14)

$$\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 = \|\mathbf{n}_j - \mathbf{n}_i\|^2 \tag{4.36}$$

to arrive at the equation

$$\kappa^2 + \tau^2 = \frac{\phi_{\mathrm{N}}}{\Delta s^2}.$$

As in to Section 4.4.2, we can assume

$$0 < r_{\min} \leq \|\mathbf{r}_{ij}\|.$$

Since the $\Delta s > \|\mathbf{r}_{ij}\|$, we have the relation

$$0 < r_{\min} < \Delta s.$$

The bound on the arc length bounds the sum $\kappa^2 + \tau^2$

$$\left(r_{\min}^2 < \Delta s^2\right) \;\Rightarrow\; \left(\kappa^2 + \tau^2 < \frac{\phi_{\mathrm{N}}}{r_{\min}^2}\right).$$

Thus minimizing the co-normality potential minimizes the sum of the squared curvature and squared torsion measures

$$(\phi_{\mathrm{N}} \to \min) \;\Rightarrow\; \left(\kappa^2 + \tau^2 \to 0\right).$$

4.4.5 Discussion

We have shown that a combination of co-planarity and Lennard-Jones minimizes the squared curvature of space curves defined between particle pairs. Likewise, a combination of co-normality and Lennard-Jones minimizes the sum of the squared curvature and squared torsion measures. We did not find a similar proof for the co-circularity potential. The co-circularity potential's goal is to encourage symmetric curvature between particle pairs, but in isolation this does not imply minimum curvature. Smooth surfaces can be constructed using only the co-planarity and co-normality potentials, and in fact many of the examples in the dissertation were created using only these two potentials.

However, in the finite element analysis of Appendix E we show that the co-circularity potential corresponds to an energy measure based on the variation in curvature of a three particle patch. And we also show that an approximation of the Gaussian curvature over the patch can be written as a sum of the co-circularity and co-normality potentials. Thus, this analysis shows that the co-planarity potential is not necessary needed to write a curvature-based energy measure. At first sight, these two different analyses may suggest contradicting results. The observation we draw is that either the co-planarity or the co-circularity potential is required, but not both. In fact the two potentials are closely related. The co-planarity potential is based on the scalar product of particle normal and the separation vector, and the co-circularity potential is based on the scalar product of the separation vector and sum of particle normals. Another point to note is that the different analyses are minimizing different curvature measures.

To summarize, minimizing co-planarity and co-normality minimizes the magnitude of curvature and torsion of curves defined between particle pairs, and minimizing co-circularity and co-normality minimizes the magnitude of the Gaussian curvature integrated over a three particle patch. That is

$$\begin{aligned} \phi_{\mathrm{P}} + \phi_{\mathrm{LJ}} &\Rightarrow \kappa_d^2 \\ \phi_{\mathrm{N}} + \phi_{\mathrm{LJ}} &\Rightarrow \kappa_d^2 + \tau^2 \\ \phi_{\mathrm{C}} + \phi_{\mathrm{N}} + \phi_{\mathrm{LJ}} &\Rightarrow \kappa^2. \end{aligned}$$

4.5 Dynamics

For practical considerations we limit particle interactions between nearest neighbors. To do this we use a distance based weighting function $w(\mathbf{r}_{ij})$ which decays the energy potential to zero at the neighborhood boundary. To control the bending and stiffness characteristics of our deformable surface, we use a scalar weighted sum of potential energies

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

The first two terms control the surface's resistance to bending, and the third term controls the surface's tendency towards uniform local curvature, and the last term controls the average inter-particle spacing.

Having defined the internal energy associated with our system, we can derive its equations of motion. By weighting our potential functions the corresponding forces and torques are different than those given in Section 4.2. The differences are (1) the weighting of the original potential terms found in Section 4.2, and (2) new force terms based on the gradient of the weighting function. The derivation of the distance weighted inter-particle forces and torques are given in Appendix D and listed below. For convenience we use the short hand notation

$$\begin{aligned} w &= w(\|\mathbf{r}_{ij}\|), \\ w' &= \frac{\mathrm{d}w(\|\mathbf{r}_{ij}\|)}{\mathrm{d}\|\mathbf{r}_{ij}\|}. \end{aligned}$$

For the spatially weighted co-planarity potential, $\phi_{\mathrm{P}}(\mathbf{n}_i, \mathbf{r}_{ij})w(\mathbf{r}_{ij})$, we have:

$$\begin{aligned} \mathbf{f}_{\mathrm{P}_i} &= (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \hat{\mathbf{r}}_{ij} w' + 2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i w \\ \mathbf{f}_{\mathrm{P}_j} &= -(\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \hat{\mathbf{r}}_{ij} w' - 2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i w = -\mathbf{f}_{\mathrm{P}_i} \\ \boldsymbol{\tau}_{\mathrm{P}_i} &= -2(\mathbf{n}_i \cdot \mathbf{r}_{ij})(\mathbf{n}_i \times \mathbf{r}_{ij})w = \mathbf{r}_{ij} \times \mathbf{f}_{\mathrm{P}_i} \\ \boldsymbol{\tau}_{\mathrm{P}_j} &= 0. \end{aligned}$$

For the spatially weighted co-normality potential, $\phi_{\mathrm{N}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij})w(\mathbf{r}_{ij})$, we have:

$$\begin{aligned}
\mathbf{f}_{\mathrm{N}_i} &= \|\mathbf{n}_i - \mathbf{n}_j\|^2 \mathbf{r}_{ij} w' \\
\mathbf{f}_{\mathrm{N}_j} &= -\|\mathbf{n}_i - \mathbf{n}_j\|^2 \mathbf{r}_{ij} w' = -\mathbf{f}_{\mathrm{N}_i} \\
\boldsymbol{\tau}_{\mathrm{N}_i} &= -2\left(\mathbf{n}_j \times \mathbf{n}_i\right) w \\
\boldsymbol{\tau}_{\mathrm{N}_j} &= 2\left(\mathbf{n}_j \times \mathbf{n}_i\right) w = -\boldsymbol{\tau}_{\mathrm{N}_i}.
\end{aligned}$$

For the spatially weighted co-circularity potential, $\phi_{\mathrm{C}}(\mathbf{n}_i, \mathbf{n}_j, \mathbf{r}_{ij})w(\mathbf{r}_{ij})$, we have:

$$\begin{aligned}
\mathbf{f}_{\mathrm{C}_i} &= \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \hat{\mathbf{r}}_{ij} w' + 2\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)(\mathbf{n}_i + \mathbf{n}_j) w \\
\mathbf{f}_{\mathrm{C}_j} &= -\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \hat{\mathbf{r}}_{ij} w' - 2\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)(\mathbf{n}_i + \mathbf{n}_j) w = -\mathbf{f}_{\mathrm{C}_i} \\
\boldsymbol{\tau}_{\mathrm{C}_i} &= -2\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)(\mathbf{n}_i \times \mathbf{r}_{ij}) w \\
\boldsymbol{\tau}_{\mathrm{C}_j} &= -2\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)(\mathbf{n}_j \times \mathbf{r}_{ij}) w.
\end{aligned}$$

For the Lennard-Jones potential we have:

$$\begin{aligned}
\mathbf{f}_{\mathrm{LJ}_i} &= -\nabla_{\mathbf{x}_i} \phi_{\mathrm{LJ}} = \hat{\mathbf{r}}_{ij} {\phi_{\mathrm{LJ}}}' \\
\mathbf{f}_{\mathrm{LJ}_j} &= -\nabla_{\mathbf{x}_j} \phi_{\mathrm{LJ}} = -\hat{\mathbf{r}}_{ij} {\phi_{\mathrm{LJ}}}' = -\mathbf{f}_j \\
\boldsymbol{\tau}_{\mathrm{LJ}_i} &= -\nabla_{\boldsymbol{\theta}_i} \phi_{\mathrm{LJ}} = 0 \\
\boldsymbol{\tau}_{\mathrm{LJ}_j} &= -\nabla_{\boldsymbol{\theta}_j} \phi_{\mathrm{LJ}} = 0.
\end{aligned}$$

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 first 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 line up the normals. The force term for the co-normality potential arises from the weighting function and can usually be ignored. 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. The Lennard-Jones force exerts a force parallel to the separation vectors holding the particle system together. It effectively counteracts the forces due to the gradient of the weighting function.

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

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

Note that most forces and torques are doubled, i.e., actions generate opposite reactions. The main 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

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

although the results would be the same. The second exception is due to the co-circularity potential which results in torques of equal magnitude but about different axes; $\boldsymbol{\tau}_{\mathrm{P}_i}$ is about the axis $(\mathbf{n}_i \times \mathbf{r}_{ij})$ and $\boldsymbol{\tau}_{\mathrm{P}_j}$ is about the axis $(\mathbf{n}_j \times \mathbf{r}_{ij})$, which may be different. However it does produce the desired result.

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

$$\mathbf{f}_i = \sum_{j \in \mathcal{N}_i} \mathbf{f}_{ij} + \mathbf{f}_{\mathrm{ext}}(\mathbf{x}_i) - \beta_0 \mathbf{v}_i - \sum_{j \in \mathcal{N}_i} \beta_1 (\mathbf{v}_j - \mathbf{v}_i) \tag{4.37}$$

$$\boldsymbol{\tau}_i = \sum_{j \in \mathcal{N}_i} \boldsymbol{\tau}_{ij} - \beta_2 \boldsymbol{\omega}_i - \sum_{j \in \mathcal{N}_i} \beta_3 (\boldsymbol{\omega}_j - \boldsymbol{\omega}_i) \tag{4.38}$$

where $\mathcal{N}_i$ are the neighbors of i, $\mathbf{v} = \dot{\mathbf{x}}$ and $\boldsymbol{\omega} = \dot{\mathbf{q}}$. 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 forces $\beta_0 \mathbf{v}_i$ and $\beta_2 \boldsymbol{\omega}_i$ and the relative velocity dependent damping forces $\beta_1(\mathbf{v}_j - \mathbf{v}_i)$ and $\beta_3(\boldsymbol{\omega}_j - \boldsymbol{\omega}_i)$. The above force equations are related to the equations of motion (3.1) and (4.1) given in Section 4.1.2. The translational damping coefficient γ_i in (3.1) is a function of β_0 and β_1. The rotational damping coefficient ξ_i in (4.1) is a function of β_2, and β_3.

4.6 Summary

We have presented a new distributed model of surface shape based on particle systems, differential geometry, and physics. We extended the volume based particle model of Chapter 3 to create oriented particles whose minimum energy configurations are sheets of particles rather than tightly packed clusters of particles. Each particle represents a local frame, a combination of position and orientation information. The particle's local XY plane represents the surface tangent plane at the particle's position, and the local Z axis represents the surface normal. Each frame can be thought of as a small surface element or discrete surface sample.

To encourage particles to arrange into surfaces, we introduced three new potential energy functions. The co-planarity potential encourages neighboring particle tangent planes to align. The co-normality potential encourages neighboring particle normals to align, controlling the "twist" in the surface between two particles. The co-circularity potential encourages surfaces of constant curvature, e.g. a sphere, rather than surfaces of zero curvature, e.g. a plane. We have also used differential geometry (Appendix A) to show how these potentials minimize the squared curvature for curves defined between particles. In the limit of zero inter-particle spacing, this is equivalent to minimizing the squared normal curvature over the surface. We also showed the co-normality potential minimize the sum of squared curvature and torsion measures for curves defined between particles. In Appendix E we show how the sum of the co-circularity and co-normality potentials minimizes the Gaussian curvature of a triangular patch defined by three particles.

Forces and torques on the particles are derived from the change in energy with respect to changes in the particle's position and orientation, respectively. In addition the surfaces respond to external forces such as gravity and collisions with other objects. In Chapter 8 we present results from simulations which illustrate the behavior of oriented particles.

Chapter 5

Continuous Descriptions

While a particle system representation of shape is sufficient for many applications, continuous surface descriptions are the standard method of describing shape in computer graphics. This chapter focuses on generating continuous surface descriptions from particle systems. First, we discuss methods used to generate surface descriptions from volumetric particle systems. Second, we discuss methods used to generate surface descriptions from particle systems of surface elements. These are different problems requiring different solutions.

5.1 Surface Descriptions Based on Volume Samplings

5.1.1 Implicit Representation

For volume based particle samplings, an *implicit* formulation of the surface provides a compact mathematical description. An implicit surface is defined as the locus of points that obey a point classification function, such as $f(x, y, z) = 0$. Assuming there exists a scalar field function that varies throughout space, an *iso-surface* is the locus of points in space whose scalar field value equals a given constant. When the field function varies continuously in space and without discontinuities, the function defines a set of closed continuous 3D surfaces. An iso-surface of threshold T is easily defined by an implicit function f as

$$f(x, y, z) = F(x, y, z) - T = 0,$$

where $F(x, y, z)$ is the value of the field in space.

To define an iso-surface equation for a particle system, we assign a continuous field to each particle. The field is maximum at the particle's center and monotonically decreases as a function of distance from the particle. The surface comprises all points in space for which the sum of the particle fields equals a threshold constant. Formally we write the iso-surface equation f as

$$f(x, y, z) = \sum_i g_i(x, y, z) - T = 0, \tag{5.1}$$

where g_i is the field function for particle i and T is the threshold value.

Blinn (1982) introduced such a class of algebraic surfaces based on control points and exponential field functions. The exponential field function results in smooth surfaces, but because the individual fields extend to infinity, it is expensive to compute the field for a given point in space. To reduce the computational effort, bounded polynomial functions of a similar shape have been used instead (Wyvill, McPheeters and Wyvill, 1986b). However, to maintain C^1 or C^2 surface continuity, one must be careful when choosing the polynomial to use.

5.1.2 Explicit Representation

By sampling the field function in space we can generate a C^0 continuous *explicit* description of the surface. The description is a polygonal approximation of the implicit surface representation, with the benefit of being easily imported into almost all commercial and public domain software rendering packages. Most iso-surface polygonization techniques are based on four steps. First, sampling points in space. Second, categorizing the points as inside or outside of the surface. Third, determining the surface intersection points between pairs of adjacent inside/outside points. And fourth, fitting polygons to the surface intersection points.

The well known Marching Cubes algorithm of (Lorensen and Cline, 1987) samples space on a cubic grid and polygonizes each cell independently. Unfortunately, ambiguous polygonizations occur since more than one possible plane will match certain combinations of in/out vertices for a given cube. These cases can be resolved by testing additional points (Wyvill, McPheeters and Wyvill, 1986b), by applying surface coherence between adjacent cells (Baker, 1988), or by sampling and testing the vertices of a tetrahedron instead of a cube (Bloomenthal, 1988; Velho, 1990).

Higher resolution polygonal approximations come at the cost of a finer grid sampling. The majority of these samples fall in empty space and provide no direct benefit. This can be observed by looking at what happens when we double the sampling rate along each axis. The number of samples in space grows by $O(N^3)$ while the number of polygons grows at only $O(N^2)$. Adaptive polygonizations benefit from concentrating polygons in areas of high curvature while reducing the number of polygons in areas of low curvature to produce a more accurate approximation for a limited number of polygons (Bloomenthal, 1988; Velho, 1990; Hall and Warren, 1990). Alternatively, methods that spread across the surface from a known surface point minimize cost by restricting computation to sample points near the surface.

5.1.3 Direct Surface Sampling

If specific points on the iso-surface are required, one can directly sample the surface by computing ray-surface intersections. The surface points are found by combining the ray equation with the implicit surface equation (5.1) and solving for the roots. The problem is complicated by the fact that for a system of N particles, in the worst case, there may be as many as $2N$ intersections with a given ray and thus $2N$ roots to the equation. For example, imagine all of the particles lying on the X axis

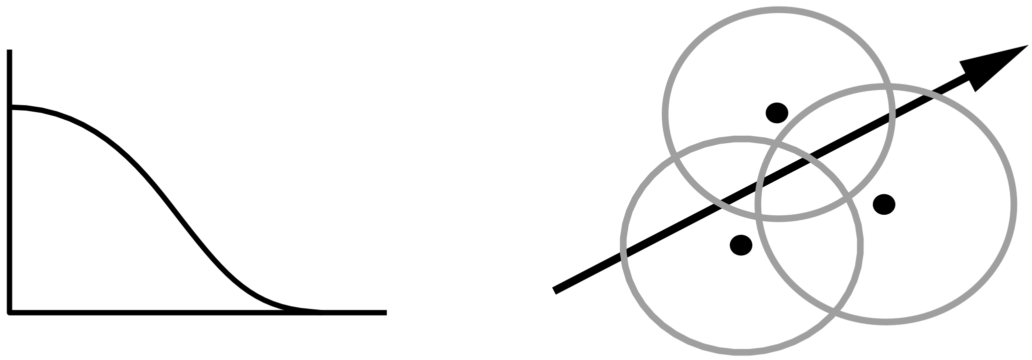


Figure 5.1: Ray tracing particle iso-surfaces

(a) A particle field function, defined as a scalar algebraic function of distance from the particle. (b) Ray intersecting field function bounding spheres. In this example the ray is split into seven non-overlapping intervals.

and spaced such that the resulting iso-surface of the field is N distinct spheres. A more likely case, and one that the potential energies encourage, is the clustering of particles into volumes, with each particle approximately r_o distance from its nearest neighbors. In this case a ray will pass through $N^{\frac{1}{3}}$ of the particle fields on average. If only the first intersection of the ray with the surface is needed, such as for rendering opaque surfaces, the problem becomes less complex. Assuming each particle field function is monotonically decreasing, one can guess an interval along the ray where the first intersection will occur (Blinn, 1982). The intersection is then isolated using an iterative root solver.

For particle fields restricted to polynomials of bounded range, there exists an algorithm to find *all* of the intersections of the iso-surface with a ray in time linear in the number of particles (Tonnesen, 1989; Wyvill and Trotman, 1989). The key idea is to split the ray into non-overlapping intervals such that each interval is represented by a single continuous algebraic equation. Figure 5.1(a) shows the graph of particle's field function as a function of distance from the particle position. A scalar field F in R^3 is defined as the summation of individual particle field functions. Since the individual field functions decay to zero at a fixed distance each separate particle field is bounded by a sphere. Figure 5.1(b) shows a simple 2D example of particles, their bounding circles, and a ray. In the 3D case, the iso-surface must lie within the union of all bounding spheres. The intersection of the bounding spheres partition space into regions, such that in each region the scalar field F is defined by the summation of a subset of the algebraic equations defining the particle fields. The intersection of a ray and the bounding spheres split the ray into non-overlapping intervals. Solving for the roots of the combined ray-field equation, over the interval, yields the intersection points of the ray and surface within that interval. For low order polynomials, computing the analytical solution is considerably faster than an iterative root finding approach. Another benefit of this approach is that it is suitable for constructive solid geometry modeling systems, which require all ray/surface intersections.

5.2 Surface Descriptions Based on Surface Samplings

We now consider the problem of generating surface descriptions from particle systems of surface samples. This is the *surface reconstruction problem.* In particular we want the surface function to *interpolate* the data, passing through the data points. This is in contrast to a surface function which approximates the data, passing close to but not necessarily through the data points. The latter functions are appropriate when the data points may not necessarily be lying on the surface, such as when there is noise in the sampling process, and the desire is to hide the noise through a smoothing process.

Reconstruction methods are classified as either *global* or *local* in nature. In a global technique, any given surface patch is dependent upon all of the data. In a local technique, any given surface patch is only dependent on nearby data points. In highly structured data, such as data sampled over a grid, the neighborhood relationships are known a priori. In our case, the case of unstructured point data, we must determine the neighboring particles. Franke (1982) observes if the data are scattered, one must inspect, in some way, all of the data to determine which points are nearby. The question arises as to whether there is such a thing as a "local" method for scattered data. Since the nature of a local surface fitting or interpolation function does not depend on how the neighbors are found, only that they are found, we may consider this to be a rhetorical question. However it reminds us that nearest neighbors must be found. The calculation of nearest neighbors is discussed in Chapter 7.2.

We have designed our particle systems so that areas of surface will be represented by collections of evenly spaced sheets of particles. Our first reconstruction goal is to reconstruct a surface which interpolates the particle positions. Our second reconstruction goal is to limit surface reconstruction to areas where the particles are sufficiently close, where closeness is defined by a distance measure. In other areas, where particles are sufficiently far apart, the goal is for the reconstructed surface to be discontinuous, with breaks or holes.

Surface reconstruction is simplified from the general case if the structure of the original surface is known. By surface structure we mean the geometric relationship between points on the surface. For example, if a set of position or point data maps uniquely to a parameterized surface such as a plane or sphere, that surface imposes a global structure over the data, and the reconstruction problem reduces to finding the best local structure that matches the global structure.

The reconstruction problem is much harder when the topology and genus is unknown, as in our case. Neither are we assuming an open or closed or surface, but which ever matches the given surface samples the "best". Since we cannot assume a global structure, we must instead be able to derive the structure from properties intrinsic to the data. One approach is to use divide and conquer, solving for local structure everywhere over subsets of the data and then combining the results to create the global structure. By choosing sufficiently small regions of shape with respect to the curvature, we can assume a locally planar structure thereby reducing the sub-

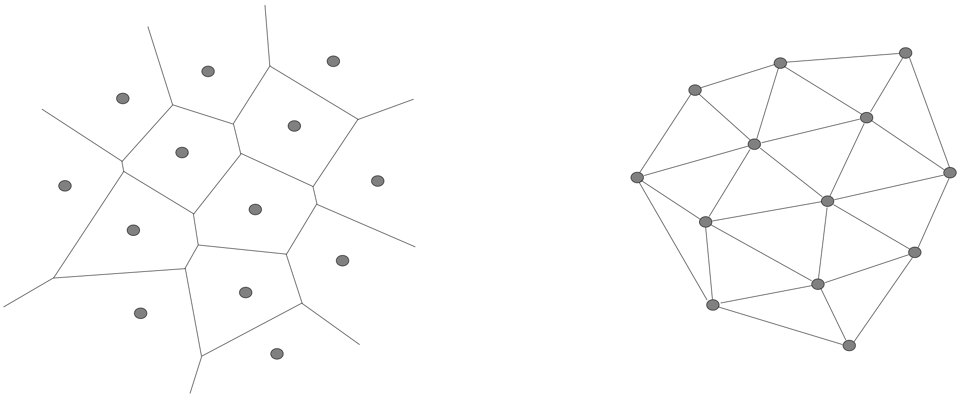


Figure 5.2: Voronoi diagram and Delaunay triangulation

(a) Voronoi diagram in 2D. (b) Delaunay graph (triangulation) as the straight line dual of the Voronoi diagram.

problem to the two dimensional case. We start by considering structure over localized areas of the shape.

5.2.1 Local Structure

An obvious property of a continuous surface is that points over a small patch are in close spatial proximity to each other. Thus it is natural to define structure based on the spatial proximity relationship between points. We consider the two dimensional problem first. The Voronoi diagram[1] can be used to solve a number of proximity problems. In particular, it solves the "loci of proximity problem". For two dimensions this is defined as (Preparata and Shamos, 1985):

> Given a set S of N points in the plane, for each point p_i in S, find the locus of points (x, y) in the plane that are closer to p_i than to any other point of S.

The solution is a partitioning of the plane into regions of spatial proximity to the original data points (Figure 5.2a). Computing the Voronoi diagram requires $O(N \log N)$ optimal computation time.

While the Voronoi diagram provides a structure over the data, it is not suitable as a surface description, since it contains unbounded edges in 2D. A better solution is to define a triangulation over the points which encodes the same proximity information. Such a triangulation is the *Delaunay triangulation*, the dual of the Voronoi diagram. A Delaunay triangulation is the graph embedded in the plane obtained by adding a straight-line segment between each pair of points whose Voronoi polygons share

[1]Voronoi diagrams also go by the names Dirichlet regions, Thiessen polygons, Wigner-Seitz cells, and proximal polygons.

an edge[2]. It is also defined as the unique triangulation such that the circumcircle[3] of each triangle does not contain any other point of S in its interior. Figure 5.2(b) shows such a triangulation constructed from the Voronoi diagram in Figure 5.2(a). The Delaunay triangulation can be computed from the Voronoi diagram in $O(N)$ time, or directly in $O(N \log N)$ optimal time. Computationally, both approaches are equivalent (Preparata and Shamos, 1985, pg. 217).

Other suggested criteria for producing good two dimensional triangulations are:

- maximizing the minimum interior angle,
- minimizing the roughness measure over a height field, and
- minimizing the total length of edges.

Intuitively, a good triangulation should avoid long thin triangles in favor of triangles that are "more or less equilateral". More formally, the criterion is to *maximize the minimum interior angle* of the triangles over all possible triangulations (Lawson, 1977). Lawson goes on to show that the max-min angle criterion, the circle criterion, and the straight-line dual of the Voronoi diagram produce equivalent triangulations in the plane.

Related to 2D triangulation is the triangulation of height field data. Rippa (1990) has suggested that a good triangulation of height field data is one that corresponds to the intuitive concept of the "smoothest surface". He defines a roughness measure as the L^2 norm squared of the gradient of the triangulation. If g_i is the planar interpolating surface for a triangle T_i then the roughness measure of T_i is

$$\int_{T_i} \left[\left(\frac{\partial g_i}{\partial x}^2 \right) + \left(\frac{\partial g_i}{\partial y}^2 \right) \right] \mathrm{d}x \ \mathrm{d}y,$$

and the roughness measure of the entire triangulation is the sum of the roughness measures of all triangles. He then goes on to show that *minimizing the roughness measure* of the triangulated height field is equivalent to the Delaunay triangulation in the plane. It is interesting to note that while the triangulation minimizes the roughness, it is also independent of values defining the height of the surface.

Another criterion for 2D triangulation is to select triangles of short edge length over longer edges. The *minimum-weight triangulation* is the triangulation that exhibits the minimal total length over all triangulation edges. At one time it was conjectured that the Delaunay triangulation was a minimum-weight triangulation, until this was disproved by Lloyd (Preparata and Shamos, 1985). Our particle systems do not present the general case of random point samples, because they naturally arrange into hexagonal configurations of nearly equal spacing and nearly equal angles between neighboring points. Thus for our systems, we expect the minimum weight and the Delaunay algorithms to generate similar if not identical triangulations.

[2]As stated this is true when there are not four or more co-circular points. However, the triangulation of these cases is straightforward.

[3]The circumcircle of a triangle is the circle such that each vertex is on the perimeter of the circle. There is exactly one circumscribing circle for a given triangle.

Two graphs related to the Delaunay triangulation are the Euclidean minimum spanning tree and the Gabriel graph. Given N points $\{p_1, p_2, p_3, \ldots, p_n\}$ in the plane, the *Euclidean minimum spanning tree* is the tree of minimum total edge length whose vertices are the given points. The *Gabriel graph* is the graph such that there is an edge between p_i and p_j if and only if the circle with diameter $\|p_i, p_j\|$ centered midway between p_i and p_j does not contain any other point. These graph structures have the following hierarchy (Preparata and Shamos, 1985; Goodman and O'Rourke, 1997).

$$\mathrm{MST} \subseteq \mathrm{GG} \subseteq \mathrm{DT}$$

For surface reconstruction of 3D point sets, Hoppe et. al. (1992) propagate orientation information by traversing the MST embedded in the graph generated by the k-nearest neighbors of each point. The constrained Delaunay triangulation has been used for triangular mesh refinement over curved surfaces (Chew, 1993). Chew extends the circumcircle test to points embedded in a curved surface, thus generating a closed space curve embedded in the surface. If no points lie inside this loop then the triangle in question is a valid triangle.

The spatial proximity defined the Voronoi diagram would appear to be a natural basis on which to form a surface interpolating our particle system. The 2D Delaunay triangulation encodes information found in the Voronoi diagram as triangles, 2D geometric simplices, and thus can provide such a surface description. In addition the Delaunay triangulation has been shown to provide well shaped triangles and minimizes the roughness of surfaces defined as height fields. It also has an elegant description, the circumscribing circle test. For our purposes, the main drawback of 2D triangulation algorithms is that they are limited to reconstructing surfaces that uniquely map to the plane.

5.2.2 3D Structure

The Voronoi diagram and Delaunay graph naturally extend to higher dimensions. The dual of an D-dimensional Voronoi diagram is the D-dimensional Delaunay graph which partitions space into volumes bounded by D-dimensional simplices[4]. For N points, there are N regions in the Voronoi diagram. Unfortunately the number of items necessary to describe the Voronoi diagram (and the Delaunay triangulation) grows exponentially with the dimension (Preparata and Shamos, 1985, pg. 246). In the three dimensional case, the Voronoi diagram partitions space into N regions which are bounded by $O(N^2)$ edges and vertices in the worst case. Its dual, the three dimensional Delaunay graph is an $O(N^2)$ collection of tetrahedra tessellating the interior of the convex hull[5].

[4] A simplex in D dimensions is a spatial configuration determined by $D+1$ points. A three dimensional simplex is a tetrahedron, a pyramid with four triangular faces.

[5] The convex hull is the smallest polyhedron such that all the points are contained within the volume. As a physical analogy, the convex hull can be thought of as applying shrink wrap plastic around a set of points. After the plastic shrinks to be a tight surface, the plastic surface is the convex hull.

While the 3D Delaunay triangulation defines a complete set of proximity relationships for a set of points, it does not provide a surface description, but rather a volumetric description of the data. We know that embedded in the Delaunay triangulation is a polyhedron which passes through all of the data points and is thus an interpolating surface. The problem now becomes: "How can we extract from this full set of proximity relations, a set of relationships which describe the surface?". If all points are on the convex hull, the convex hull is the surface. If not, one can remove simplexes from the set of tetrahedrons until all the points are on the boundary. Boissonnat (1984) solved this for surfaces of genus 0 (without holes) by providing a set of rules to iteratively remove simplices with a face on the boundary. The remaining tetrahedrons describe the shape's volume and the boundary faces define the surface. To my knowledge, this approach has not been extended to the general case of manifolds of arbitrary genus.

Alpha shapes were designed to allow scientists to explore the spatial structure of points sets, by extracting subsets of the Delaunay triangulation (Edelsbrunner and Mücke, 1994). Given a omnipresent ball with radius α, a polytope (a face or edge) of the triangulation is removed when the ball can enclose the polytope without enclosing any of the vertices. The resulting triangulation is the alpha shape. Surfaces can be reconstructed from point sets by extracting surfaces embedded in α-shapes (Guo, Menon and Willette, 1997), though there is a trade off between reconstructing over sparsely sampled areas and maintaining details in densely sampled areas. Unfortunately all of these techniques are based on first generating the 3D Delaunay triangulation, at a worst case time cost of $O(N^2)$ and expected cost of $O(N)$ (Dwyer, 1991).

5.2.3 Triangulation of Particles

Applying two dimensional surface reconstruction solutions suffice for small areas, but does not solve the larger three dimensional problem. Generating three dimensional proximity information does not simplify the problem, but transforms the problem into one of extracting a surface description from a volumetric description. Conceptually we would like the favorable qualities of a 2D triangulation to extend to triangulating our shape in 3D. One approach would be to locally compute a 2D triangulation by projecting the subsets of data onto a plane, triangulating the data points, and then projecting the triangulation back onto the original data set. Difficulties arise in determining what are appropriate subsets of points to consider and how to merge the resulting triangulations together. The real problem is that our data defines full three-dimensional shapes of unknown structure and not special cases which reduce to the two-dimensional domain. The N-dimensional Delaunay triangulation tests for inclusion or exclusion of N-dimensional simplices. In 2D it tests triangles. In 3D it tests tetrahedrons. We propose a test that takes the criterion of a two dimensional Delaunay triangulation and extends it into the three dimensional domain while preserving the two dimensional test.

In 2-D, a triangle is part of the Delaunay triangulation if no other vertices are within the circle circumscribing the triangle. To extend the circumscribing circle idea to 3-D, we check the smallest sphere circumscribing each triangle. This is also the 3D analogue of the 2D Gabriel graph. Given any three points, if no other points fall within the smallest circumscribing sphere, then these three points define a valid triangle. If another point lies within the sphere, the triangle is not part of the surface triangulation. Note that the smallest sphere circumscribing three points has embedded in it the smallest circle circumscribing the three points, which is the 2D Delaunay test. This circle is a great circle of the sphere.

To avoid computing triangles in areas without particles, we limit the length of valid triangle edges (to 2 units of inter-particle spacing, by default). This also has the side effect of pruning the number of triplets to test from $O(N^3)$ to $O(N)$. The total cost of the triangulation is $O(N \log N)$; $O(N \log N)$ to search for all nearest neighbors and $O(N)$ to test for valid triangles. The reconstruction heuristic works well in practice when the surface is adequately sampled with respect to the curvature. To better visualize the resulting surface, Gouraud, Phong, or flat shading can be applied to each triangle. The results of using our triangulation algorithm are shown in Figure 5.3, where the original point set is shown along with the resulting triangulation displayed in wireframe and randomly colored filled triangles. Figure 8.17(e) is an example of a Gouraud shaded triangulation.

5.2.4 Undesirable Triangulations

The goal of the triangulation algorithm is to compute surface connectivity based on the spatial proximity of particles. However, the arrangement of the particles may not suggest a “reasonable” surface. For example, consider the four vertices of a regular tetrahedron (i.e. a tetrahedron made of equilateral triangles). From this set of vertices, there are four combinations of three vertices that one can choose. Each of these triplets corresponds to the face of the tetrahedron and each of these triplets will pass the minimum sphere test. Thus, given these four vertices, the triangulation algorithm will generate the faces of a regular tetrahedron. For this case, this is a reasonable surface to generate.

Now let us consider the case of two spherical arrangements of particles, say $\mathbf{V}_1$ and $\mathbf{V}_2$. Suppose that when $\mathbf{V}_1$ and $\mathbf{V}_2$ are far apart, the triangulation algorithm generates two triangulated spherical surfaces. Now further suppose $\mathbf{V}_1$ and $\mathbf{V}_2$ are moved closer together, such that two vertices from $\mathbf{V}_1$ and two vertices from $\mathbf{V}_2$ correspond to the vertices of a regular tetrahedron. It is possible that the triangulation algorithm will generate two spherical arrangements of triangles, plus the faces of a tetrahedron and hence connect the two sets of triangles together. This may be considered an “undesirable” triangulation of the points. We should note that generally such an arrangement of particles would not be a minimum energy configuration for an oriented particle system, unless there are external forces acting on the system.

As the two sets of particles are brought even closer together it is likely (depending on the full set of circumstances) that the two spherical arrangements of particles will merge, much like two soap bubbles will join with a wall between the two bubbles. For particles interacting under the influence of the Lennard-Jones, co-circularity, and co-planarity potentials, this would be a valid minimal energy configuration. Barring

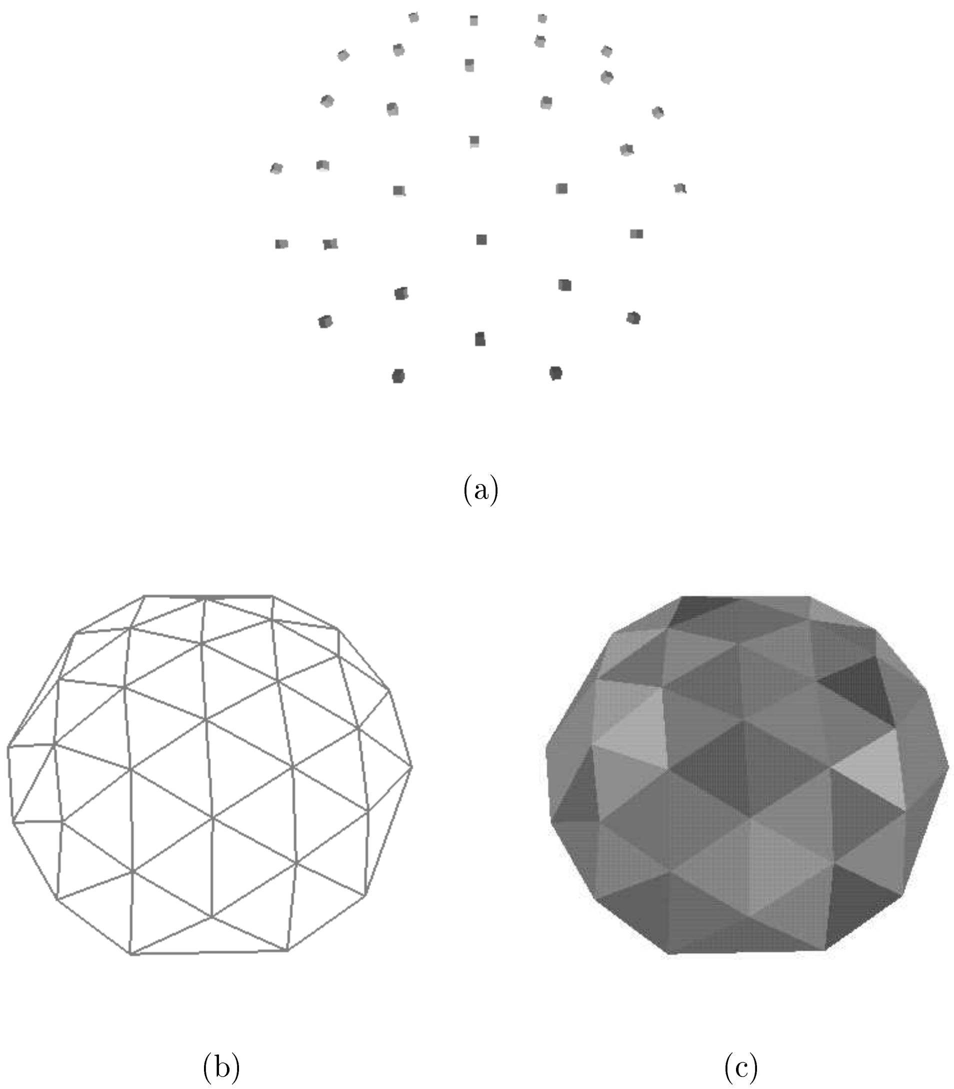


Figure 5.3: Triangulation of points

(a) A subset of points taken from a sphere of particles at equilibrium. (b) Triangulation displayed as wireframe. (c) Triangulation displayed as filled triangles of various colors.

large external forces, the co-normality potential could be used to prevent collections of particles with opposing normals from merging together. Such arrangements of particles are the collision of two spheres (as above) and the folding of a sheet of particles onto itself. With an appropriate choice of maximum triangle edge length, this may prevent the generation of triangles between the two sheets.

Since our triangulation algorithm imposes no constraints on the possible positions of the particles, we do not guarantee that the triangles generated will appear as a "reasonable" or "properly" connected surface. Given a set of samples from a known surface, it would be interesting to prove the limits of surface curvature and sampling density over which our triangulation algorithm can correctly reconstruct the surface. We leave this as an open problem.

We end this section on a philosophical note. The real question facing us is not, "Does the triangulation algorithm guarantee reasonable surfaces?", but rather the more general question, "What is reasonable surface connectivity for *arbitrary* sets of points?".

5.2.5 C^1 Continuity

For generating smooth surfaces from polygonal meshes, one can apply surface subdivision methods to replace the original polygon mesh with successively finer polygon meshes, which in the limit results in a curvature continuous smooth surface (Catmull and Clark, 1978; Peters and Reif, 1997) except at a number of extraordinary points (Doo and Sabin, 1978). Many subdivision schemes produce meshes which are a combination of quadrilaterals and convex polygons, with extraordinary points introduced for polygons which are not quadrilaterals and at vertices with edge valence not equal to four. Unfortunately, when applied to our triangulations, these approaches will introduce a high number of extraordinary points: approximately $2N$ for Peter and Reif's scheme and $3N$ for Catmull and Clark's scheme. Also, many of the polygons are slow to refine. Luckily there exist subdivision surfaces designed for triangulated surfaces of arbitrary topology. For example, Loop (1987) defines an approximating subdivision schemes specifically for triangular meshes. His method generates refined triangular meshes at each iteration and the extraordinary points are limited to the original mesh vertices. Zorin (1996) defines a subdivision scheme well suited for triangulations. In the limit, it interpolates a C^1 smooth surface between the original vertices .

For imaging purposes, increasing the resolution of the particle system can be as effective as fitting a C^1 surface. As the area of each triangle approaches the area of an image pixel, the visual difference between a C^0 surface and the corresponding C^1 surface becomes negligible. With the ability to render over 1 million anti-aliased texture mapped triangles per second on current low end graphics workstations to 80 million on high end systems[6], rendering large quantities of triangles is a feasible alternative.

[6]Statistics from SGI's Silicon Surf world wide web site (October 1997) discussing the SGI *O2* and *Onyx2 RealityMonster* systems.

For guaranteed continuity, there are spline based techniques for specifying C^1 and G^1 surfaces over polyhedra of arbitrary genus. Most techniques require the polyhedron to be a triangulated polyhedron, thus circumventing difficult cases. In the general case, the problem reduces to transforming a polyhedron into a triangulation. In our case this is not a problem, since our polygonization is already a triangulation. Guo and Menon (1996) and Bajaj and Ihm (1992) interpolate triangulations of arbitrary topology with implicit spline patches. Both assume normal vectors are defined at the vertices, and construct intermediate geometries as a precursor to constructing the surface patch. Loop (1994) presents an algorithm for approximating a triangular mesh of arbitrary topology with triangular surface patches that meet with G^1 continuity. The drawback of such techniques is that the generation of high order polynomial patches which are expensive to compute. A second, equally significant drawback in many applications (such as character animation) is the lack of control compared to hand crafted piecewise spline surfaces.

5.3 Summary

This chapter discussed methods of generating continuous surface descriptions from the discrete description provided by our particle system. We break the continuous surface description problem into two separate problems, one for volume particles and one for surface particles.

For particle systems of volume elements, we recommend an implicit surface approach to surface description. In this model, each particle is associated with a monotonically decreasing field in R^3 as a function of distance from the particle. This field corresponds to the volume of the particle. The summation of all particle fields defines another scalar field in R^3 which is a scalar field defined for the particle system, and iso-surface is defined by the locus of points in R^3 equal to a constant scalar "threshold" value. For a given point in space, testing the value of the field at that point against the threshold value computes whether the point is inside, outside, or on the surface. Polygonal approximations to the iso-surface can be constructed by sampling the field function on a regular grid and determining surface intersections between pairs of adjacent inside/outside points. Direct surface sampling techniques, such as ray-tracing, can also be used for rendering. For general field functions, iterative root solving techniques can be used to find the surface-ray intersections. For field functions defined by low order polynomials, analytic solutions can be found over discrete segments of the ray.

For particle systems of surface elements, we present an algorithm to construct triangulations over an even distribution of particle samples. Our algorithm is based on spatial proximity information as encoded in the Voronoi diagram and its dual, the Delaunay graph. However, neither 2D nor 3D Delaunay tests are directly applicable to our problem. In 2D the three point circumscribing circle test identifies valid triangles, and in 3D the four point circumscribing sphere test identifies valid tetrahedrons. Instead we wish to identify valid triangles in 3D. To do so we extend the essence of the 2D test to 3D. If no other points fall within the smallest sphere circumscribed by three points, these three points define a valid triangle of our surface. Note that in this test, the smallest circle circumscribing the three points (the 2D test) is a great circle of the sphere. To allow our surfaces to separate, we limit the lengths of triangle edges. This has the secondary advantage of reducing the number of particle triplets to test to be linear in the number of particles. For generating smooth surfaces from our triangulations, either subdivision surfaces or interpolating triangular spline patches can be used.

Chapter 6

Thermal Energy

Due to the nature of the Lennard-Jones energy function, it can be used to model both the limited flexible deformations of a solid and the rapidly varying geometry of a liquid. By varying the dissociation energy of the Lennard-Jones function, we can model a continuous range of materials. For rigid solids, we increase the magnitude of the dissociation energy, and for flexible solids we decrease the magnitude. Further decreases result in fluid like behavior. Thus by varying the dissociation energy as a function of thermal energy, we can create models that "melt" and "freeze". This chapter describes in detail how we model the effects of thermal energy on the potential energy function, and a model of the continuous heat equation in terms of discrete particles.

6.1 The Heat Equation

At the macroscopic level, the thermal energy ψ in a body A is given by integrating over the volume,

$$\psi = \int_V \rho\sigma\theta \, dV = \iiint_A \rho\sigma\theta \, dxdydz, \tag{6.1}$$

where V is the volume, $\rho(x, y, z)$ is the mass density of the body, $\sigma(x, y, z)$ is the specific heat, and $\theta(x, y, z)$ is the temperature. The amount of heat leaving a body per unit time is given by

$$\int_S \mathbf{n} dS = \iiint_A \nabla \cdot (K\nabla\theta) \; dxdydz, \tag{6.2}$$

where S is the surface, $\mathbf{n}$ is the surface normal, and K is a 3×3 symmetric matrix known as the thermal conductivity matrix.

Setting the rate of decrease of thermal energy in the body equal to the amount of thermal energy leaving the body, we arrive at the partial differential equation called the heat equation

$$\frac{\partial\rho\sigma\theta}{\partial t} = \nabla \cdot (K\nabla\theta) \,. \tag{6.3}$$

For a homogeneous and isotropic material $K = kI$, where I is the identity matrix, the equation reduces to the familiar form

$$\frac{\partial \rho\sigma\theta}{\partial t} = k\nabla^2\theta, \tag{6.4}$$

where ∇^2 is the Laplacian.

6.2 Discrete Heat Equation

We use a discrete approximation of the general heat equation (6.3) to solve for the thermal energy and temperature of each particle over time. The thermal energy of a particle is related to the temperature as follows. We assume the specific heat σ and temperature θ are constant for a given particle. The mass m of an object is equal to integrating the mass density ρ over the object's volume. Since we already know the mass, we do not need to specify the mass density or volume of a particle. From (6.1) the thermal energy ψ_i for a particle reduces to

$$\psi_i = \sigma_i\theta_i m_i. \tag{6.5}$$

The change in thermal energy of particle i over a time interval Δt

$$\frac{\psi_i^{t+\Delta t} - \psi_i^t}{\Delta t} \tag{6.6}$$

approximates the left side of the heat equation.

To approximate the right side of the equation, the $\nabla \cdot (K\nabla)$ term, we introduce a thermal conductivity variable k_{ij} between each pair of particles i and j. The thermal conductivity is a measure of the rate of thermal energy transfer within a given material. Insulators, such as Styrofoam, will have a lower thermal conductivity than conductors, such as steel. We compute the approximation on a pairwise basis

$$\nabla \cdot (K\nabla\theta) \approx \sum_{j\in\mathcal{N}_i} \frac{\frac{1}{n_i}k_{ij}\left(\theta_j - \theta_i\right)}{\frac{1}{4}r_{ij}^2}, \tag{6.7}$$

where r_{ij} is the distance between particles i and j, $\mathcal{N}_i$ is the set of nearest neighbors for particle i, and n_i is the cardinality of $\mathcal{N}_i$.

The approximation is based on the finite difference method. To see this, we look at the 2D hexagonal configuration of particles shown in Figure 6.1. Let us consider the case of the center particle, numbered 0, which has 6 neighboring particles, numbered 1 through 6. The Laplacian of the temperature at the center particle $i = 0$ is

$$\nabla^2\theta_0 \approx \frac{\frac{1}{6}\left(\theta_1 + \theta_2 + \theta_3 + \theta_4 + \theta_5 + \theta_6\right) - \theta_0}{\frac{1}{4}r^2} = \sum_{j=1}^{6} \frac{\frac{1}{6}\left(\theta_j - \theta_0\right)}{\frac{1}{4}r^2} \tag{6.8}$$

when approximated by finite differences (Vitasek, 1969). By combining (6.6) and (6.7), we arrive at a discrete version of the heat equation over a three dimensional hexagonal grid. Heat dissipation into the external environment can easily be modeled by adding a term such as $-\beta_h\theta_i$ to the discrete heat equation.

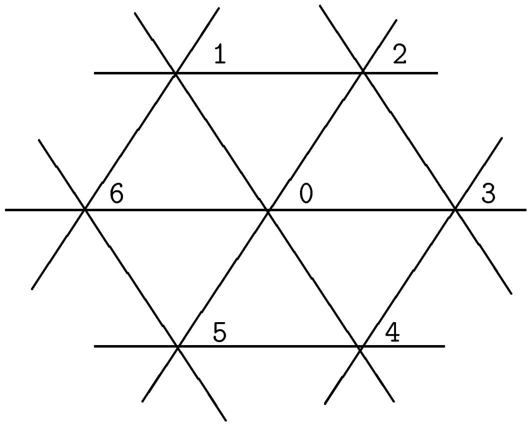


Figure 6.1: Finite difference grid for the Laplacian

6.3 States of Matter

Having laid the basic ground work, we now consider the physical states of matter as a discrete system. Liquids are qualitatively different from solids and the difference is a matter of geometry (Barton, 1974). At the molecular level, when external forces are insignificant, molecules arrange into closely packed structures to minimize their total energy. For spherically symmetrical potential energy functions in 2-D, particles arrange into hexagonal orderings (Figure 3.5). In 3-D, the molecules arrange into hexagonal ordered 2-D layers.

As thermal energy is added to the system, the molecules begin to vibrate and the entropy in the system increases. This movement is quantified at the molecular level as kinetic energy. It is the relationship between molecular kinetic energy and inter-molecular binding energy that determines the amount of entropy, and thus the state of the ensemble. At low temperatures, the mean binding energy is greater than the mean kinetic energy, and the material is highly ordered as a solid. At high temperatures the mean kinetic energy is much greater than the mean binding energy and the material is in total disorder as a gas. In the liquid state the molecules are sufficiently close together for there to be local ordering, with only a small amount of compressibility, but the kinetic energy is large enough to prevent long-range ordering. It is the ability to provide such fluid changes in structure that we which to capture in our approach to modeling.

6.4 Thermal Energy

As an object heats up, we do not increase the kinetic energy of the particles as Greenspan does in his simulations of solids (Greenspan, 1973) or as in the molecular model (Trevena, 1975). Instead, we collapse the mean kinetic energy (thermal energy) of a small volume into a change in potential energy between volumes. We model thermal energy and the inter-particle potential energies together by a continuous

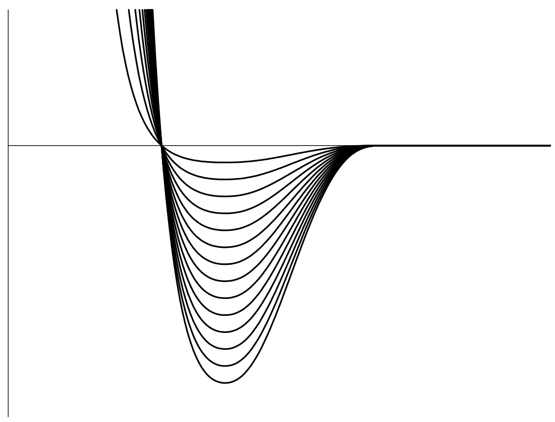


Figure 6.2: Family of inter-particle potential functions.

The weighted Lennard-Jones potential function plotted for $m = 2$, $n = 4$, $r_o = 1$, $r_a = r_o$, $r_b = \sqrt{3} r_o$, and $e = 1 \ldots 14$. The potential is weighted to include only nearest neighbors.

---

family of inter-particle potential energy functions. Figure 6.2 shows several functions from a family of functions. "Cold" temperatures map to functions with low potential energy minima (deeper energy wells), while "hot" temperatures map to functions with higher potential energy minimal (shallow wells). As thermal energy is added to a system the total energy increases accordingly.

It is instructive to compare the differences between the typical molecular dynamics model using particles and our particle approach. In the molecular dynamics model, the addition of thermal energy increases the kinetic energy and the inter-molecular potential energy function is fixed. The average speed of a volume of molecules corresponds to a measure of thermal energy. The average velocity of a volume corresponds to the rigid body motion of that volume.

In our model, the addition of thermal energy decreases the inter-particle potential energy while the kinetic energy is unchanged. The velocity of a single particle directly parallels the rigid body motion of a small volume of material, i.e. the average velocity of a molecular volume. The dissociation energy of the Lennard-Jones potential corresponds to the combination of the binding properties and thermal energy. Addition of thermal energy increases the total energy of the system *without directly changing particle velocity*, thus avoiding the numerical instabilities, and possible visual artifacts, that would be introduced by increasing kinetic energy. We next discuss how we modify the binding potential. We discuss the practical consequences in the summary (Section 6.7).

6.5 The Thermoelastic Lennard-Jones Function

We now derive a thermoelastic version of the Lennard-Jones potential energy. To insure conservation of energy, we impose the following condition on our system: the addition of an amount of thermal energy to a particle system should increase the total energy of the system by that amount of energy, while leaving the kinetic energy of the system constant. Let us consider some given particle system in two different states, an unheated state $\mathbf{S}_0$ and in a heated state $\mathbf{S}_1$. We let E_{S0} be the energy of the system with zero thermal energy, and E_{S1} be the energy of the same system with the addition of E_H thermal energy. We write the energy conservation condition as

$$E_{S0} + E_H = E_{S1}.$$

Expanding the system energies into kinetic and potential components by (3.9) we have

$$E_{K0} + E_{P0} + E_H = E_{K1} + E_{P1}. \tag{6.9}$$

To derive thermal energy in terms of potential energies, we must formulate the problem in terms of particle pairs. Toward this end, we make several simplifying assumptions. These are:

- particle-particle interaction is limited to nearest neighbors,
- the system is at equilibrium,
- and thermal energy in E_1 is uniform throughout the system.

The first assumption can be easily enforced by setting our weighting function to go to zero at the range of the second nearest neighbor, a distance of $\sqrt{3}r_o$. The second assumption is equivalent to saying the system begins at absolute zero, and thus it must be in equilibrium. Because it is at equilibrium and interaction is including only nearest neighbors, the particles will be separated by the equilibrium spacing.

We begin by rewriting the potential energies E_{P0} and E_{P1} in terms of dissociation energy and thermal energy. The term E_{P0} is equal to the summation of the pairwise potential energies of an unheated system

$$E_{P0} = \sum_{i}^{N} \sum_{j \in \mathcal{N}_i} w_{ij} \phi_{ij\,0}, \tag{6.10}$$

where $\phi_{ij\,0}$ is potential defined for zero thermal energy. The magnitude of dissociation energy of the Lennard-Jones function for zero thermal energy is equal to some constant value, say e_0. Since the system is near equilibrium, neighboring particles are separated by the equilibrium separation r_o, the weighting function w_{ij} evaluates to unity, and the potential energy $\phi_{ij\,0}$ evaluates to $-e_0$. Thus the inner summation reduces to a product of the average number of neighbors h times the dissociation energy e_o

$$\sum_{j \in \mathcal{N}_i} w_{ij} \phi_{ij\,0} = \sum_{j \in \mathcal{N}_i} (1)(-e_0) = -he_0.$$

and (6.10) reduces to

$$E_{P0} = -Nhe_0. \tag{6.11}$$

Now let e_1 be the dissociation energy between a pair of particles in the heated system $\mathbf{S}_1$. Similar to above, the potential energy of $\mathbf{S}_1$ is

$$E_{P1} = -Nhe_1. \tag{6.12}$$

We can now compute a formula for the unknown dissociation energy e_1. Combining (6.9), (6.11), and (6.12) results in

$$E_{K0} - Nhe_0 + E_H = E_{K1} - Nhe_1.$$

By noting that the kinetic energy is constant, that is $E_{K0} = E_{K1}$, this reduces to

$$E_H = -Nh(e_1 - e_0).$$

Since we assume thermal energy is added uniformly to the system, we divide the thermal energy E_H among the N particles assigning each particle ψ thermal energy

$$N\psi = -Nh(e_1 - e_0).$$

The dissociation energy for zero thermal energy e_0 is constant and we solve for e_1,

$$e_1 = e_0 - \frac{\psi}{h}.$$

This describes a method to vary the Lennard-Jones dissociation energy allowing us to account for thermal energy.

To allow for heat transfer throughout the system, we remove the assumption of uniform addition of thermal energy and allow the particles to have differing amounts of thermal energy. We modify the dissociation energy defined between two particles to be a function of the average of both particles, and two constants; the dissociation energy between a pair of particles with zero thermal energy e_o, and the average number of neighbors h (6 for a surface, and 12 for a volume).

$$e_{ij}(\psi_i, \psi_j) = e_o - \frac{\psi_i + \psi_j}{2h}$$

The final form of the thermoelastic Lennard-Jones potential is as follows:

$$\phi(\|\mathbf{r}_{ij}\|, \psi_i, \psi_j) = \frac{-1}{m-n}\left(e_o - \frac{\psi_i + \psi_j}{2h}\right)\left(m\left(\frac{r_o}{\|\mathbf{r}_{ij}\|}\right)^n - n\left(\frac{r_o}{\|\mathbf{r}_{ij}\|}\right)^m\right), \tag{6.13}$$

where $\|\mathbf{r}_{ij}\| = \|\mathbf{x}_j - \mathbf{x}_i\|$ is the magnitude of separation between particles i and j. This formulation is valid for all values of $\psi < e_o h$. Note that negative values of temperature are equivalent to increasing the stiffness of the material, i.e. increasing the magnitude of e_o.

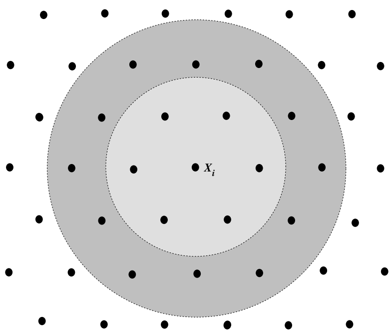


Figure 6.3: Neighborhood Interactions

6.6 Thermal Expansion

When the temperature of a body is raised or lowered, the material will expand or contract. Thermal linear expansion is described by the following formula (Case, 1938)

$$l_2 = l_1 \left(1 + \alpha(\psi_2 - \psi_1)\right) \tag{6.14}$$

where l_1 is the length of a rod at temperature ψ_1, l_2 is the length at temperature ψ_2, and α is the linear expansivity. That is, the rod will expand by the amount $l_1\alpha\Delta\psi$. If the expansion is opposed, then the resulting stress $El_1\alpha\Delta\psi$ will be determined by Young's modulus (3.19). Linear volume expansion is described by

$$V_2 = V_1 \left(1 + \beta(\psi_2 - \psi_1)\right)$$

where V_1 is the volume at temperature ψ_1, V_2 is the volume at temperature ψ_2, and β is the volume or bulk expansivity. For values of less than 2%, the volume expansivity is approximated well by $\beta = 3\alpha$. In general, solids and liquids expand upon heating, and contract upon cooling [CRC mathbook pg. 317]. The expansivity coefficients for liquids tend to be much larger than for solids. Water is a notable exception in which the expansivity depends on temperature and has a negative value over the range of 0^oC and 4^oC.

The thermal energy model for our particle system maintains a common equilibrium separation and collision distance, to avoid unwanted side effects. The expansion or contraction of the particle model due to temperature changes is dependent on the neighborhood range. We consider two cases for which the system is at minimum energy.

In the first case, the neighborhood range is such that a given particle only interacts within the first shell of neighboring particles as shown by the smaller circle in Figure 6.3. At equilibrium, each neighboring particle will be separated by the equilibrium separation r_o and the inter-particle forces will be zero. As the object heats, the equilibrium separation will remain constant and the object neither expands nor contracts. To tailor the equilibrium separation to specific materials, one could vary the equilibrium separation according to (6.14), rewritten here as

$$r_o(\psi_2) = r_o(\psi_1) \left(1 + \alpha(\psi_2 - \psi_1)\right) \tag{6.15}$$

where α is the linear expansivity, $r_o(\psi_1)$ is the equilibrium separation at ψ_1, and $r_o(\psi_2)$ is the equilibrium separation at ψ_2. Coefficients of expansion for a wide variety of solids and liquids can be found in (Miner and Seastone, 1955).

In the second case, the neighborhood range is expanded to include particles beyond the nearest neighbors, i.e. particles within the larger circle shown in Figure 6.3. At equilibrium, there will be compression of the solid due to the long range attractive forces between the center particle and the distant neighbors. As the temperature decreases, these attractive forces will become stronger and the particle system will contract further. In this case, the model mimics the general rule of decrease in volume when the temperature decreases and expansion of volume when the temperature increases.

6.7 Summary

The contributions of this chapter are the introduction of a particle based thermoelastic model with energy conservation and the introduction of a discrete heat transfer model derived from the macroscopic heat equation. Our thermoelastic model states that the elastic force of the Lennard-Jones potential is linearly related to the thermal energy (and thus temperature), as is the thermoelastic model presented by (Terzopoulos, Platt and Fleischer, 1989) for spring-mass systems. It extends their model to particle systems and adds the constraint of conservation of thermal, kinetic, and potential energy over the system. Heat transfer is based on the continuous heat equation at the macroscopic level, which is implemented as a finite difference scheme over a hexagonal grid.

The model includes the physical quantities of thermal energy, volume, mass density, mass, specific heat, temperature, and thermal conductivity. For each particle we need to add only one new variable of state, the thermal energy of the particle, and one variable to accumulate the heat transfer between neighbors. A thermal conductivity variable could also be added to each particle to model non-homogeneous materials. The remaining physical quantities can be defined as constants or derived from existing state variables.

Our model focuses on variations in malleability and does not model a variety of more complex phenomena found in real world materials. For example, it does not necessarily model the changes in volume found in some materials such as the expansion of water upon freezing, the shrinking of many metals when cooled and some plastics upon heating. Instead, the model encourages conservation of volume. Such volumetric changes could be added to the model by defining the equilibrium separation parameter r_o as a function temperature, as given in equation (6.15). Our model is valid for the range $\psi < he_o$ which accounts for solid and fluid behavior. It is invalid when $\psi \geq he_o$. That is, we do not model gaseous behavior.[1]

[1]When $\psi = e_o h$, the Lennard-Jones potential evaluates to zero, and when $\psi > e_o h$ the potential inverts, resulting in long range repulsive forces and short range attractive forces. The long range repulsive forces mimic the nature of a gas. The short range attractive forces encourage nearby particles to occupy the same position, unlike a gas. Thus if we wanted to mimic gaseous behavior,

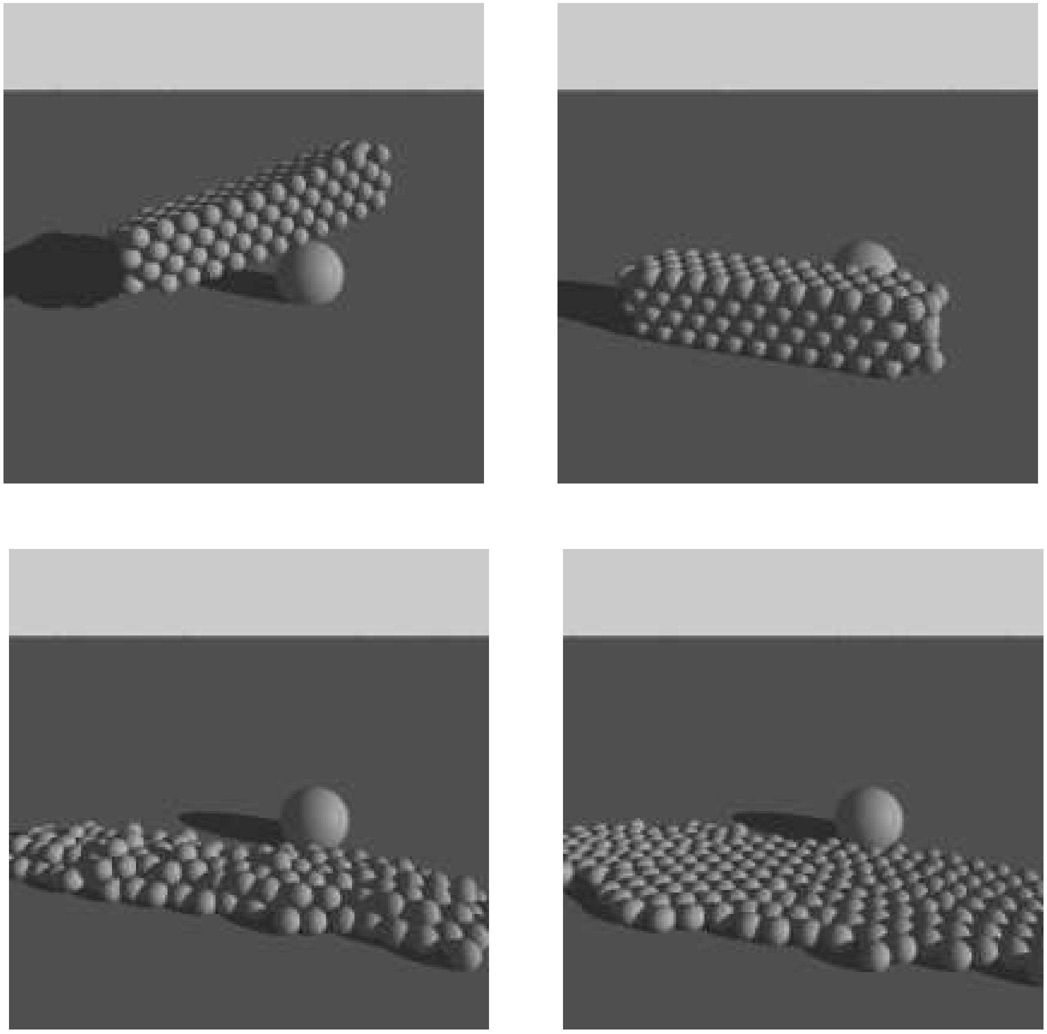


Figure 6.4: Beam colliding and melting

The first two frames shows the initial rigid structure of the beam colliding with an object and coming to rest on the ground plane. It is highly ordered and maintains it's structure even under external forces and collisions. As heat is transferred from the ground plane into the object, the inter-particle bonds weaken and the beam exhibits fluid like behavior with quickly changing structure.

A simple example of melting is shown in Figure 6.4. A solid beam is dropped, colliding with a sphere and the ground plane. After coming to rest, it is "heated" by the ground plane. Heat is transferred through the beam according to the discrete heat transfer model. The resulting weakening of the inter-particle bonds causes the solid to "melt", thus losing its initial structure. Other applications of heat are shown in Figures 8.11 and 8.12, where heat is used to locally modify the properties of a surface, making it more malleable than the surrounding material.

we would need to replace the attractive force with a repulsive force for all values $\psi \geq e_o h$. One possible solution would be to replace the Lennard-Jones with a purely repulsive force at these values.

Chapter 7

Implementation Issues

In this chapter we review issues relevant to the implementation of dynamically coupled particle systems. Of importance are the issues of efficiently computing the forces for large numbers of particles, computing the set of nearby neighbors for each particle, the stable and efficient numerical integration of the computed forces over time, and methods of interactively visualizing the system.

7.1 Efficient Force Computations

The state of a dynamically coupled particle system at a given time is described by the set of particle positions, orientations, translational and angular velocities. Using this state information, the forces and torques on each particle can be computed directly from the inter-particle force functions. By numerically computing a discrete approximation of the force over the time interval, the system state can be updated to the next time step.

The tractability of these computations is a limiting factor in the ability to interact with these systems in real-time. With appropriate assumptions we can reduce the complexity in both time and memory requirements. In this section we discuss the computational problems, possible solutions, and our choices.

7.1.1 Direct Pairwise Computation

The definition of the force on a particle i due to the other particles, is the sum of the pairwise inter-particle forces

$$\mathbf{f}_i = \sum_{j \neq i}^{N} \mathbf{f}_{ij},$$

where N is the number of particles. A straight forward computation of the forces for the above equation is conceptually and computationally simple, but not necessarily the most efficient. Such an evaluation will require $O(N^2)$ operations, making it reasonable for small systems, yet prohibitively expensive for large values of N. Since we may use on the order of several thousand particles we need a better solution.

7.1.2 Mesh Methods

When individual inter-particle forces are small, yet the cumulative effects of inter-particle forces are significant, then a mesh method may be useful in reducing the amount of computation (Hockney and Eastwood, 1988). If the forces can be described in terms of a continuous potential field, then a mesh approach is valid. The basis of the approach is to approximate a field that is continuous in space by a set of discrete values on a finite mesh covering of the space.

The total cost of the mesh approach is proportional to both the number of particles and the number of mesh points. Thus, for a coarse grid, the mesh method may result in large gains of speed over the direct approach. One drawback is that the gain in speed is traded for a loss of accuracy, as each step in the process introduces new errors. In addition, the accuracy is highly dependent upon the relationship between the continuous functions and the mesh spacing. In order for the mesh to adequately approximate a system, the mesh spacing must be smaller than the important wavelengths of the system. That is, potential fields are poorly represented on the mesh for distances less than the mesh spacing. Mesh techniques become useful when the problem involves smoothly varying forces.

To approximate the Lennard-Jones force would require a mesh spacing significantly smaller than the average inter-particle spacing, resulting in more mesh points than particles. Thus, for our problem, the mesh approach will consume more time than the direct $O(N^2)$ approach.

7.1.3 Combined Methods

The third approach combines features of the direct and mesh approaches, taking advantage of their respective strengths (Hockney and Eastwood, 1988; Greengard, 1988). It is adequate for *highly correlated systems* with smoothly varying long-range forces. The key of the approach is to split the inter-particle forces into two parts:

$$\mathbf{f} = \mathbf{f}_s + \mathbf{f}_l \tag{7.1}$$

a rapidly varying short range force $\mathbf{f}_s$ which is nonzero for only a few inter-particle distances, and the slowly varying long range force $\mathbf{f}_l$ which is sufficiently smooth to be accurately represented on a mesh. The direct method is used to find the total short-range force contribution on each particle, while the mesh method is used to find the total long-range contribution. The total cost is proportional to the number of particles and the total number of mesh points. This is a reasonable approach for the potential energies we are using.

7.1.4 Limited Force Range

The final approach is to compute the short range forces using the direct particle-particle computation, and to ignore distant forces. This is a valid assumption in highly correlated systems where there exists strong short range inter-particle forces and the distant forces are insignificant. Assuming a reasonably distributed set of particles, there will be at most a small number of neighboring particles that contribute to the total force of a given particle, resulting in $O(N)$ computation time for the entire system of N particles.

The drawback of this approach is that it is restricted to forces that decay to near zero at the force cut-off boundary, otherwise discontinuities will be introduced into the continuous force function. It is wise to avoid such discontinuities as they will create instabilities when we numerically integrate the forces.

7.1.5 Discussion

From considering the choices available to us, the combined method and the direct method[1] with a limited force range are the two computationally feasible approaches. The mesh method is not suitable for highly correlated systems such as ours, and the full direct method would require $O(N^2)$ operations per time step, thereby limiting interactive shape modeling to small systems of particles.

The nature of our applications suggests the use of the direct method with limited force range as the best solution. We have designed our inter-particle potential functions so that the forces decay as a function of distance, allowing distant forces to be ignored. Still the potential discontinuity at the boundary introduces a force discontinuity contributing to numerical instabilities. To alleviate this we can use the weighting function (3.12) which goes to zero at the boundary, thus insuring a continuous force function. In addition to computational efficiency considerations, limiting the potential functions provides a consistent potential energy representation of the merging and splitting characteristics of our dynamically coupled particle systems. When two objects are separated by the force range distance, the potential energy description of the two objects is also independent.

An important consideration, we have neglected so far in our analysis, is the computational cost required to find the neighbors of each particle. We discuss this next.

7.2 Neighbor Computation

We now look at the problem of finding the set of neighboring particles for each particle. This problem is formalized as the range search problem.

7.2.1 The Range Search Problem

An essential condition of computing the inter-particle forces is finding the neighbors of each particle. This can be stated as finding an $O(N)$ subset of particle pairs from the $O(N^2)$ possible pairs, such that each selected pair is no farther apart than a specified distance r. This type of search problem, known as the *range search* problem, has been studied extensively in the computer science literature (Overmars, 1983; Preparata and

[1]The names "direct", the "mesh", and "combined" methods also go by the names of "particle-particle", "particle-mesh", and "particle-particle-particle-mesh (P^3M)" respectively(Hockney and Eastwood, 1988).

Shamos, 1985; Samet, 1989; de Berg et al., 1997; Goodman and O'Rourke, 1997). Formally we state this as:

> Given a set of three-dimensional points $\{x_1, x_2, \ldots, x_N\}$, and a sphere of radius r, centered at position $\mathbf{x}_s$, which points $\mathbf{x}_i$ satisfy the condition $\|\mathbf{x}_s - \mathbf{x}_i\| \leq r$, where $\|\mathbf{x}_s - \mathbf{x}_i\|$ is the Euclidean distance measure?

A single range query is easily performed in linear time, by examining each of the N points. Linear space will suffice as only the positions must be stored. However, to compute the force of the system, we will need to find the nearby neighbors of each particle, resulting in $O(N^2)$ operations. In addition to the computing forces, the range search problem appears in triangulation, particle creation heuristics, and in the use of modeling tools. Similar to our desire to reduce the force complexity to something tractable, we want to reduce the computational complexity of the range search problem.

The approach to choose depends upon the application at hand and properties of the data, such as: Are the range queries of a constant size search radius or variable search radius? Is the data set fixed, or will points be added and deleted? Are the data points uniformly distributed throughout space, or highly localized?

We consider the approach to use in light of the application areas of this thesis: computer assisted animation, geometric modeling, and surface reconstruction. Below we list the qualities of the various applications, while noting that the needs of specific applications within these fields may vary. For all applications, the range queries will generally be of a small range for the force, triangulation, and particle creation heuristics. Likewise we can expect the use of modeling tools to range from including all of the particles to only a few. For all applications the points will be uniformly distributed in space at the local level, though non-uniform at the global level. Volumes will have denser local grouping of particles than will surfaces. For all application areas we can expect cases when we will want to add or delete particles from the system. The basic search operation required, in order of frequency, are: (1) range search, (2) insertion, (3) deletion, and (4) point existence. The last three operations assist in interactively grabbing, moving, adding, and deleting particles.

In the remainder of this section, we review several types of spatial data structures and comment on them in relation to our dynamically coupled particle system. Because our points sets and queries will be relatively well behaved, we will focus our discussion on simple data structures that will perform well on average, instead of focusing on complex data structures optimized for worst case performance. We begin with the hashing, followed by direct access grid structures, and end with multi-resolution tree structures.

7.2.2 Hashing

Hashing is a process used to quickly distribute and retrieve data among a fixed number of memory locations. Given a data point $\mathbf{p}_i$, a hash function $H(\mathbf{p}_i) = A_i$ can be used to compute an address A_i which is the address of a block of reserved memory, called a *bucket*, in which the data point is stored for later reference. Frequently this is implemented using low level bit operations to provide fast index computation. It has become common usage to define all indexing functions which use low level bit operations as hash functions. An example of such a function[2] is given in Graphic Gems (Glassner, 1993)[page 343]. Instead, we will use the theoretical definition of hashing as given below.

The theoretical goal of hashing is to take highly structured and sparse data in the domain, and provide a mapping into a uniformly distributed range, thus reducing the amount of memory required while providing constant time access. This is normally accomplished by defining the hash function to be a randomizing function with uniform distribution. A given input to the hash function must always produce the same output address in order to store and subsequently retrieve data values. Thus the function cannot be truly random, but rather pseudo-random. A random and uniform distribution of data, has the advantage of reducing the amount of memory required to be of the same order as the size of the data set. A second advantage is the hash function provides direct access to each data item for efficient insertion, retrieval, and deletion. There are, however, two corresponding disadvantages (Samet, 1989). First, it is difficult to find a suitable hash function that will map each data point to a unique location. Second, since hashing functions randomize the data, the function destroys the spatial relationships between data points. Destruction of the spatial relationships makes such a hashing scheme unsuitable for solving the range search problem.

7.2.3 Fixed Grid

The fixed grid is a spatial data structure that partitions space into equal constant size cells, with each cell storing the points falling within the cell volume. By partitioning 3D space along orthogonal planes, the mapping from a bounded R^3 volume to the finite number of cells can be directly computed from each point's x, y, and z values. The extent of the 3D grid can be easily computed by finding the minimum and maximum values of x, y, and z values. Provided the point set is uniformly distributed, the fixed grid can provide constant time insertion, deletion, point existence, and range search queries of small search radii. Thus it is an appealing solution to our problem.

The performance of the fixed grid depends on the ratio of the grid volume search to the actual search volume, the bucket size, and the distribution of points in space. As the cell width decreases, a smaller volume of the grid is searched, but at the cost of accessing more cells. A standard cell width to choose is one equal to the search radius. This requires testing the cell in which the center of the search radius falls, and the adjoining cells, for 27 cells in all. Using half the cell width results in accessing five times as many cells and approximately half the total volume. Using double the cell width results in accessing only eight cells, but almost three times the total volume. The most efficient search will result in a tradeoff between accessing more cells and testing fewer points. The optimal balance between the cell width and search volume depends on the relative costs of the accessing more cells and of testing a point for

[2] This function concatenates the high order bits of three integer indices together. The result is a new integer index.

inclusion. Both are implementation dependent. The number of points in an average cell is also dependent on the distribution of particles. Due to the short range repulsive and long range attractive forces, particles will be uniformly distributed at a local level, in sheets for surfaces and in tightly packed clusters for volumes. At the global level, the shape of the object determines the distribution. In general, we suggest using a cell width equal to the search radius.

The amount of memory required to represent the grid is dependent upon the number of cells. The number of cells is a function of the extent of the grid and the cell width. We begin by considering a simple example, such as modeling a sphere. If we model the sphere by volumetric particles, the number of cells in the grid will $O(N)$ where N is number of particles. If we model the sphere by surface particles, the number of cells in the grid will be $O(N^{3/2})$. This indicates that for large systems of oriented particles the majority of the grid will not be used. A 2D example of this is shown in Figure 7.1(a). Of course, we could construct cases where the solid model is sparse in space, such as a long thin twisted wire. In this case the grid will grow faster than linear in the number of volumetric particles. Likewise, we could construct cases where a surface has numerous folds effectively filling space. In this case the number of grid cells grows linearly with the number of surface particles. From a practical perspective we will assume dense solid models and surfaces that do not tend to fill space. Thus we will state that *in general* the amount of memory used will be of $O(N)$ for volume particles and $O(N^{3/2})$ for surface particles.

The final consideration is the bucket size. When the bucket size is too large, memory resources will be wasted. When the bucket size is too small, bucket overflows will occur requiring additional computation. To find a balance we consider the distribution of the points in space. Let us consider the two simple cases: (1) when particles are modeling a volume, and (2) when the particles are modeling a surface.

Volume Case

In the volume case, the particles are uniformly distributed throughout the volume of the object model. Due to the Lennard-Jones force, at equilibrium the particles will be separated by the equilibrium separation r_o, and under external forces we will assume no closer than the collision distance σ. To find an upper bound on the number of particles in a cell we will assume the cell is completely enclosed by the model volume, with particles represented by tightly packed spheres of diameter σ. From the volume packing density (Section 3.4.1) we can compute the number of tightly packed particles that will be present in a given volume, thus resolving the issue of bucket size. The upper bound on the number of particles in a cell is the integer ceiling of

$$P_v \frac{V_1}{V_2} = \sqrt{2} \left(\frac{w}{\sigma} \right)^3$$

where P_v is the volume packing factor for tightly packed spheres (3.20), V_1 is the volume of a cubic cell of width w, and V_2 is the volume of a sphere of diameter σ.

Surface Case

We now consider the uniform distribution of particles over the surface of a model. We could consider the case of a planar section cutting through the cell, but to create a better upper estimate we consider a surface of high curvature. Let us assume the cell encloses a hemisphere with diameter equal to the cell width. From the area packing density, we compute the number of particles in the cell as the integer ceiling of

$$P_a \frac{A_1}{A_2} = \frac{\pi}{\sqrt{3}} \left(\frac{w}{\sigma}\right)^2$$

where P_a is the area packing density for tightly packed circles (4.1.9), A_1 is the area of the hemisphere of diameter w, and A_2 is the area of a circle with diameter σ.

Summary

In summary, the grid provides fast constant time insertion, deletion, point existence, and range search queries for our problem when the optimal cell and bucket sizes are chosen as described above. We suggest a cell width equal to the search radius for efficient range searching. When modeling surfaces, in general, the number of grid cells will grow $O(N^{1/2})$ faster than the number of particles, suggesting this is not the best structure for surfaces. However when modeling volumes, in general, the grid grows linearly with the number of particles making it a reasonable choice.

7.2.4 Reduced Grid

For certain classes of data, we can combine aspects of the fixed grid data structure and hashing to create a "reduced grid" which limits the amount of memory required while maintaining fast storage, retrieval, and range searching (Szeliski, 1998). For this discussion we relax our definitions of the fixed grid and of hashing. We use the definition of hashing as the ability to quickly compute an index value and instead of randomizing the data, the hashing function will maintain the spatial structure inherent in the data set. The definition of a grid is modified so that cells map to multiple disjoint regions of the volume.

One reduces the number of cells in a grid of size M^3 by a factor of D^3 by applying a modulo based hashing function when computing the grid indices for each point. If the fixed grid indices for a point are the integers i, j, and k, then the reduced grid indices for this point would be $i \bmod E$, $j \bmod E$, and $k \bmod E$, where $E = M/D$. When E is a power of two, this is equivalent to masking out the high order bits of the index. One can also think of this as a redistribution of points through a modulo transformation of coordinate space. The result of this transformation is being able to reduce the number of cells necessary for storage, while keeping the original spatially close points in adjacent cells, where adjacency includes the concept of "wrapping around" the grid. A range search on the reduced grid will be guaranteed to return spatially close points, yet it may also returned distant points. A second drawback, is that point sets which are aligned with the major axes or point sets with

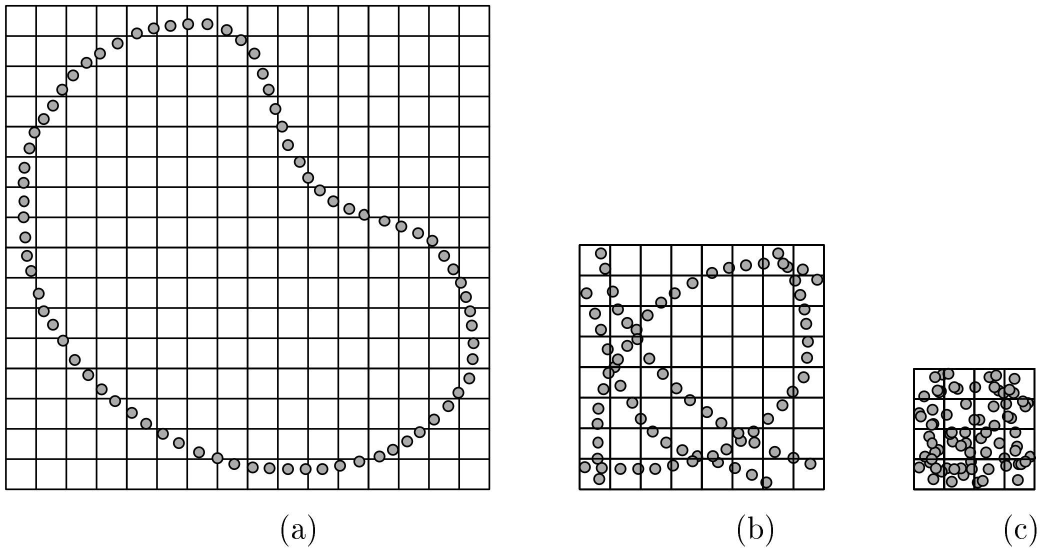


Figure 7.1: Storage of points in a grid.

(a) A 2D example a string of points stored in a 16x16 fixed grid. (b,c) Same points stored in reduced grid using modulo based indexing. (b) Grid reduced to 8x8, (c) Grid reduced to 4x4.

periodic structures may create high density regions in the reduced grid. Despite these drawbacks, reducing memory requirements by several orders of magnitude may be a sufficient reason for its use in certain applications. The reduced grid is illustrated in Figure (7.1) with a hypothetical 2D curve formed by a string of points.

7.2.5 Hierarchical Structures

One way to reduce the total number of unused cells and avoid the the bucket overflow problem is to adaptively vary the cell size through recursive subdivision (Samet, 1989). On overflow the volume of a cell is divided among a set of smaller cells, each with its own bucket. The original cell becomes a *parent* cell, and the new cells are *children* of the parent. As a child cell's bucket becomes full, it will become a parent cell, split, and create children. The relationship between the parents and children can be described by a directed tree structure where the nodes of the graph correspond to cells and the arcs between nodes correspond to the parent-child relationships.

Region-Based Tree Structures

A *region quad-tree*[3] is a directed tree-based data structure which recursively decomposes space based on the density of points. Initially, a quad-tree starts as a single

[3]In two dimensions, a quad-tree subdivides space into four quadrants. A three dimensional quad-tree, commonly called a oct-tree, subdivides space into eight octants. The quad-tree easily generalizes to higher dimensions. For generality we will use the term quad-tree to refer to such a tree in any dimension.

cell represented by the root of the tree. As data points are added, they are stored in the root cell's bucket until it is full. When full, the cell subdivides its space among 2^d equal size children cells each with its own bucket (where d is the dimension of the space), and then transfers each of its data points to the appropriate child. Applied recursively, this creates a tree structure.

Similar to the region quad-tree, the *region kd-tree* recursively subdivides space. At each level of the tree it subdivides space into two equal size regions, alternating the axis along which space is divided. This gives the kd-tree a branching factor of two at each node, compared to 2^d for the quad-tree.

The region quad-tree and region kd-tree exhibit identical theoretical time complexity and are suitable for our purposes. Building a tree of N points will take on average $O(N \log N)$ operations and require $O(N)$ memory. Point insertion, point deletion, and point existence queries each require $O(\log N)$ operations on average, where $\log N$ is the average distance from the root node to a leaf node. A range search can be performed in $O(F + N^{1-1/d})$ in the worst case, and in practice the query will take $O(F + \log N)$, where F is the number of points returned and d is the dimension. Since we expect particles to be evenly distributed over an area, at the local level, it is unlikely that the worst case time will actually occur.

Point-Based Trees

Point-based trees are spatial data structures used to encode a set of point data, by recursively subdividing space based on the location of individual points. Each node in the tree corresponds to a unique point in the data set. For a data set with N points, this results in exactly N nodes in the tree.

According to Overmars (1983), such quad-trees were the first data structures devised for solving the range search problem efficiently. In a *point quad-tree*[4] one of the data points is taken to be the root, and based on the value of the data point, a d-dimensional space is subdivided into 2^d quadrants, thus splitting the data set into 2^d subsets. These subsets, minus the original data point, will be encoded in the subtrees of the root node. This process is recursively applied until each quadrant contains at most one data point.

The *point kd-tree* was introduced as an improvement over the point quad-tree. While the quad-tree is a 2^d-ary tree, the kd-tree is a binary tree. This reduces the branching factor at each node and thus the overall storage requirements. At each level down the tree, the point kd-tree divides the set of points in two, alternating the dimensions along which space is split.

The theoretical computational complexity of the point quad-tree and point kd-trees are the same as their region based counter-parts: $O(\log N)$ time for insertion,

[4]It has been suggested that the region quad-trees and the region kd-trees be called quad-tries and kd-tries, to differentiate them from the original point based tree structures they were derived from. However this naming convention never became popular, and at least in computer graphics the unqualified term "quad-tree" usually refers to a (region) quad-trie, and not the original (point) quad-tree. To distinguish between the two data structures (quad-trees and quad-tries), We use the term "quad-tree" preceded by the qualifiers "region" (a trie) and "point" (the original tree).

and point existence queries, $O(N \log N)$ time to build the tree, and $O(N)$ memory. A range search query takes $O(N^{1-1/d})$ in the worst case and $O(F + \log N)$ in the average case. However deletion is more complex because the data points also serve to partition the space from which they are drawn.

Range Trees

A third type of tree, the *range tree*, yields better worst-case range search times at the expense of storage and a more complicated implementation (Overmars, 1983; de Berg et al., 1997). A range tree query performs a 1-dimensional search along the first dimension to select a subset of the data. The subset of data is then searched using a 1-dimensional search along the second dimension to find a smaller subset of the data. This procedure is recursively applied until all dimensions are exhausted. Range trees require $O(N \log^{d-1} N)$ storage, $O(N \log^{d-1} N)$ operations to build, and provide a worst case range search time of $O(\log^d N + F)$. Layered range trees (de Berg et al., 1997) use fractional cascading to reduce the search query time by a factor of $O(\log N)$ to $O(\log^{d-1} N + F)$, while maintaining the same storage requirements. The worst case range search time is better than the quad-tree or kd-tree worst case, but in practice the quad-tree and kd-tree will perform as well, or better (Overmars, 1983).

Summary

Region based and point based tree structures provide an elegant approach to solving the range search problem. Point based trees tend to be simpler to code and maintain, because:

- The partitioning of space is implicit in the data points rather than explicitly specified as required by a region based tree.
- Leaf nodes and interior nodes can be the same data type.
- One does not have to maintain and search buckets of data.

However, when searching the tree for data points, a region based tree may require slightly less computation. This is because the use of buckets reduces the depth of the tree and thus the time required to find the relevant data points. One should note, however, that for certain data sets the region based trees may result in highly unbalanced trees. Kd-trees have the advantage that the algorithm is independent of the dimensionality of data, allowing one to write a single kd-tree implementation for both 2D and 3D simulations. Also a well designed implementation can be used for searching arbitrary types of k-dimensional data, not just point based data. It has also been argued that the kd-tree branching factor of two is preferable to the 2^d branching factor of a quad-tree, because this reduces the memory cost of interior nodes (Samet, 1989). From a theoretical point of view, the region based trees, the point based trees, quad-trees, and kd-trees exhibit similar computation and memory costs. Range trees provide better worst cast performance at the expense of memory and a more complicated implementation, but in practice the simpler tree based data structures will perform as well or better.

7.2.6 Discussion

A fundamental problem encountered in the implementation of dynamically coupled particle system is the problem of finding the neighbors for each particle. This problem appears in the computation of forces, in the particle creation heuristics, in surface triangulation, and in the use of modeling tools. The problem is formalized in Section 7.2.1 as the range search problem. To solve the range search problem we have compared the use of hashing, fixed grids, reduced grids, and hierarchical based data structures.

Hashing, a technique for efficiently storing and retrieving data, is ineffective for computing such spatial relationships. The fixed grid data structure, which partition space in equal size cells, is a good choice for volume modeling, though inefficient in memory for surface modeling. The reduced grid data structure overcomes the memory problem of fixed grids, though search performance may be poor for data sets aligned with the major axes and for data sets with spatially periodic structure. Since our particles will be uniformly distributed with respect to the object being modeled, hierarchical structures are good for both volume and surface based particle systems. The uniform distribution will tend to create well balanced trees resulting in efficient access. When the cell bucket size is chosen wisely, the region-based trees maybe be faster than point-based trees, as the time needed to traverse farther down a point-based tree will outweigh the extra distance tests that will be included in a region-based tree search query. If we use uniform splitting of regions, a region-based tree structure can produce the same partitioning of space as a fixed grid, but with varying resolution. Thus, we can use arguments similar to those presented for the fixed grid to select a leaf cell bucket size.

The best data structure to use will ultimately depend on the given application, and whether one wants to favor generality or speed. In practice, we use a region kd-tree with uniform splitting along each dimension. To further reduce computation, we perform this operation occasionally and cache the list of neighbors for intermediate time steps. We use the kd-tree for both our surface and volume modeling research. Our implementation is independent of the dimension of the data, the size of the data set, the data types, and the distribution of data. While we chose a kd-tree for generality, a production system may wish to trade generality for a strategy finely tuned to the problem at hand. For such a finely tuned systems, we would suggest a region tree or reduced grid for surface modeling, and a region tree, fixed grid, or reduced grid for volume modeling.

7.3 Numerical Integration

In this section we discuss methods for integrating the forces of our particle system over time to arrive at a new system state consisting of a position and orientation for each particle.

7.3.1 Equations of Motion

Having resolved our method of efficiently computing the forces, torques, and the neighbors for a system of particles, we now turn to the problem of integrating the forces through time to compute the time-varying positions and orientations. To create materials which behave as real-world materials, with momentum and the transfer of momentum between objects, we integrate a second order dynamics system. This is appropriate for applications such as physics-based animation where the motion of the objects over time, is more interesting than the final energy minimum. On the other hand, a first order dynamics system is more appropriate for many applications which solve an optimization criterion, such as surface fitting to data samples, For this case, it is adequate to assign the computed forces to the velocity vectors and solve for the change in positions. This is a straight forward simplification and not discussed further. We will discuss the more complex problem of integrating a second order dynamical system.

An un-oriented particle system is governed by (3.1). An oriented particle system is governed by (3.1) and (4.1). Including the model of heat, the inter-particle force terms are functions of the form

$$\mathbf{f}_i^{int} = \mathbf{f}_i^{int}(\mathbf{x}_1, \mathbf{q}_1, \psi_1, \mathbf{x}_2, \mathbf{q}_2, \psi_2, \ldots, \mathbf{x}_N, \mathbf{q}_N, \psi_N)$$

and the inter-particle torque terms are of the form

$$\boldsymbol{\tau}_i^{int} = \boldsymbol{\tau}_i^{int}(\mathbf{x}_1, \mathbf{q}_1, \mathbf{x}_2, \mathbf{q}_2, \ldots, \mathbf{x}_N, \mathbf{q}_N),$$

which is identical to equation (4.4). The external force and torque terms $\mathbf{f}_i^{ext}$ and $\boldsymbol{\tau}_i^{ext}$ are functions of individual particle positions, particle orientations, and external state variables and are given by the Lagrangian equations of motion (4.3), and (4.5).

An optimal solution strategy depends on the type of equations we are trying to integrate. Equations (3.1) and (4.1) are second-order non-linear ordinary differential equations. The force and torque functions introduce highly non-linear terms making this a difficult problem to solve. In addition, the torques are dependent on both particle orientations and positions, thus coupling (3.1) and (4.1) together. We characterize our problem as an initial value problem on two sets of coupled second order non-linear differential equations, with the goal of determining the values of the dependent variables $\mathbf{x}_i$ and $\mathbf{q}_i$, for $i = 1, 2, \ldots, N$, at a set of values in the independent variable t.

The standard approach is to first reduce the second order system of differential equations into two coupled sets of first order differential equations, and then numerically approximate the unknown dependent variables using a finite difference based scheme (Press et al., 1992). For example (3.1) can be rewritten as

$$\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = \mathbf{v}_i \tag{7.2}$$

$$\begin{aligned}\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} &= \frac{\mathbf{f}_i}{m_i} &(7.3)\\ \mathbf{f}_i &= \mathbf{f}_i^{int} - \mathbf{f}_i^{ext} - \gamma_i \mathbf{v}_i, &(7.4)\end{aligned}$$

where the velocity $\mathbf{v}_i$ is a new state variable. An equivalent reduction exists for (4.1) Using numerical techniques, the values of $\mathbf{x}_i$ and $\mathbf{v}_i$ can then be advanced from their values at time t to their new values at time $t + \Delta t$ for some value Δt.

7.3.2 Overview of Integration Methods

To solve the initial value problem, all numerical techniques discretize the changes in the dependent and independent variables as finite steps, e.g. $\Delta t, \Delta \mathbf{x}_i$, and $\Delta \mathbf{q}_i$. Doing so allows one to represent the differential changes as algebraic equations composed of these finite steps. As the step size is made small, a good approximation to the underlying differential equation is achieved. There are two approaches to constructing such algebraic equations. Explicit schemes produce new dependent variable values explicitly from the algebraic equations in terms of previous dependent variable values. Implicit schemes produce the new values implicitly in terms of both previous and new dependent variable values.

Implicit schemes applied to linear systems are guaranteed to be stable for any step size, and for non-linear systems are known to generally exhibit good stability. The computational complexity required at each time step is their major drawback. Implicit formulations are easily solved by matrix factorization for linear systems of equations. However, non-linear systems of equations have to be solved iteratively at each step, which can be extremely slow compared to explicit methods. Semi-implicit methods result when one linearizes a non-linear system of equations, and then iteratively solves the linearized version. This still requires matrix factorization, perhaps more than once per time step.

Writing the particle system dependencies as a matrix will result in an $M \times M$ matrix, where M is the number of particles times the number of degrees of freedom per particle. Assuming a particle system with nearby neighbor dependencies, the corresponding matrix will be unstructured and sparse. In fact, the positions of the non-zero elements will be continually changing due to the dynamic coupling inherent in our system. While there exist specialized techniques for inverting certain subclasses of sparse matrices, such as banded diagonal matrices, unstructured sparse matrices will require more general techniques. One could use standard techniques such as Gauss-Jordan elimination or LU decomposition, both requiring $O(N^3)$ time[5]. Since our sparsity pattern is irregular one may want to use matrix techniques designed for sparse linear systems (Press et al., 1992), to reduce the fill-in that occurs during matrix inversion. Such techniques have three steps: (1) An analyze step is done once for each non-zero pattern, (2) a factorize step is computed for each matrix fitting the pattern, and (3) an operate step is computed for each new right-hand side of a matrix equation. Since our particle couplings will be dynamically changing, the

[5]Matrix inversion software solutions in general require $O(N^3)$ time, however it has been shown that inversion can be computed in $O(N^{\log_2 7})$ (Press et al., 1992, page 104).

analyze portion, the most computationally expensive of the three steps, will need to be repeatedly computed along with the other two steps. Relaxation methods can also be used to speed up the convergence of a solution to a matrix equation. Multigrid relaxation methods (Terzopoulos, 1986) alone and combined with conjugate gradient descent algorithms (Szeliski, 1990) have been shown to decrease the computational time needed to find minimal energy states for optimal surface fitting problems in computer vision.

Explicit schemes are popular because new values can be computed directly from previous values. However, unless care is taken, they have a tendency to become unstable or require a small step size. There are a variety of explicit schemes to choose from depending on accuracy and efficiency. According to Hockney and Eastwood (1988),

> The compromise between accuracy and efficiency can be altered in two ways - either by using a higher-order scheme and larger time step or by using a lower-order scheme and smaller time step. The former approach suffers because (1) the time step is limited by natural frequencies of the system, (2) higher-order schemes often have more restrictive stability limits on the time step, and (3) high-order schemes need force values at several time levels. Usually, the best compromise between accuracy, stability, and efficiency in *many-body calculations* is found by using a simple second-order accurate schemes (such as leapfrog) and adjusting the time step accordingly.

7.3.3 Integration

We have chosen to integrate the system of particles using low order explicit schemes primarily because our focus has been on applying our system to a variety of problems including *interactive* surface modeling. While explicit schemes have drawbacks, implicit or semi-implicit scheme will generally be slower than linear time, making these unsuitable for interactive applications. We have used both the Euler and the leapfrog integration methods. Both approaches require a single evaluation of the force equations for a given time step.

Explicit Euler

At each time step we sum the all of the forces acting on each particle and integrate over the time interval. We solve our second order system of differential equations as two coupled first order systems. We have used Euler's method for its simplicity (Press et al., 1988). However instead of using Euler's method directly on both systems we can make a better estimate for the second system by using the average of the old and new velocities computed by the Euler step on the first system (Gould and Tobochnik, 1988). Both the Euler and the modified Euler method have local truncation error of $O(h^2)$ at each step and are first order accurate integration schemes.

We designate time values by superscripts, defining t as the current time and integrate over the time interval $h = \Delta t$ to time $t + h$. We use the equations of motion from Appendix B that are simplified for our choice of inertia tensor. The translational equations are

$$\begin{aligned}\mathbf{a}^t &= \frac{\mathbf{f}^t}{m} \\ \mathbf{v}^{t+h} &= \mathbf{v}^t + h\,\mathbf{a}^t \\ \mathbf{x}^{t+h} &= \mathbf{x}^t + h\frac{\mathbf{v}^{t+h} + \mathbf{v}^t}{2}.\end{aligned}$$

and angular motion equations are

$$\begin{aligned}\mathbf{b}^t &= \mathbf{I}^{-1}\boldsymbol{\tau}^t \\ \boldsymbol{\omega}^{t+h} &= \boldsymbol{\omega}^t + h\,\mathbf{b}^t \\ \boldsymbol{\theta} &= h\frac{\boldsymbol{\omega}^{t+h} + \boldsymbol{\omega}^t}{2} \\ \mathbf{q}_{\boldsymbol{\theta}} &:= \left[\cos\left(\|\boldsymbol{\theta}\|/2\right), \hat{\boldsymbol{\theta}}\sin\left(\|\boldsymbol{\theta}\|/2\right)\right] \\ \mathbf{q}^{t+h} &= \mathbf{q}^t\mathbf{q}_{\boldsymbol{\theta}}.\end{aligned}$$

For angular motion, the change in orientation $\boldsymbol{\theta}$ is computed from the angular velocity $\boldsymbol{\omega}$, where both are represented as vectors. The incremental rotation $\boldsymbol{\theta}$ is then converted to the quaternion $\mathbf{q}_{\boldsymbol{\theta}}$ (Shoemake, 1989), before updating the particle orientation through quaternion multiplication[6]. To avoid a loss of accuracy, due to finite floating point representation, we normalize the quaternion after each step.

Leapfrog

The leapfrog scheme is defined by the time centered finite differences

$$\frac{x^{t+h} - x^t}{h} = v^{t+h/2}$$

and

$$\frac{v^{t+h/2} - v^{t-h/2}}{h} = a^t,$$

defined here for a one-dimensional space. It is similar to the Euler method in that the same amount of work is required and no auxiliary storage is needed, as is required in higher order methods. The leapfrog scheme differs from the explicit Euler in that the velocity values are defined at the midpoint between time steps. The result is a local truncation error at each step of $O(h^3)$ making this a second order accurate integration scheme. A differential equation is time-reversible if a particle integrated forwards in time in a given force field will retrace its path and return to the starting point, when integrated backwards in time. Time-reversible difference approximations are obtained by defining time-centered derivatives, such as the leapfrog scheme (Hockney and Eastwood, 1988).

[6]Note, the vector $\hat{\boldsymbol{\theta}}$, in the vector to quaternion conversion, is the unit vector in the direction $\boldsymbol{\theta}$.

Using the same notation as for the Euler equations, the leapfrog equations for translational motion are

$$\begin{aligned}
\mathbf{a}^t &= \frac{\mathbf{f}^t}{m} \\
\mathbf{v}^{t+h/2} &= \mathbf{v}^{t-h/2} + h\,\mathbf{a}^t \\
\mathbf{x}^{t+h} &= \mathbf{x}^t + h\,\mathbf{v}^{t+h/2}.
\end{aligned}$$

The equations for angular motion are

$$\begin{aligned}
\mathbf{b}^t &= \mathbf{I}^{-1}\boldsymbol{\tau}^t \\
\boldsymbol{\omega}^{t+h/2} &= \boldsymbol{\omega}^{t-h/2} + h\,\mathbf{b}^t \\
\boldsymbol{\theta} &= h\,\boldsymbol{\omega}^{t+h/2} \\
\mathbf{q}_{\boldsymbol{\theta}} &:= \left[\cos\left(\|\boldsymbol{\theta}\|/2\right), \hat{\boldsymbol{\theta}}\sin\left(\|\boldsymbol{\theta}\|/2\right)\right] \\
\mathbf{q}^{t+h} &= \mathbf{q}^t\mathbf{q}_{\boldsymbol{\theta}}.
\end{aligned}$$

Unfortunately leapfrog does not compute v^{t+h}, but rather $v^{t+h/2}$. If v^{t+h} is needed, such as to compute a damping force, an approximation can easily be computed as follows. We construct the two formula

$$\mathbf{v}^t = \frac{\mathbf{v}^{t-h/2} + \mathbf{v}^{t+h/2}}{2}$$

and

$$\mathbf{v}^{t+h/2} = \frac{\mathbf{v}^t + \mathbf{v}^{t+h}}{2}.$$

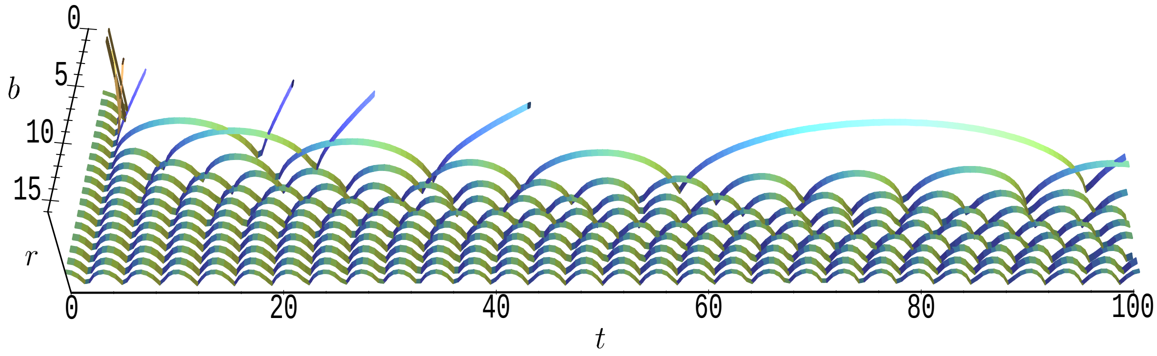
to describe the velocity at times t and $t + h/2$ as average velocities. Solving for $\mathbf{v}^{t+h}$ results in the extrapolation

$$\mathbf{v}^{t+h} = \frac{3\mathbf{v}^{t+h/2} - \mathbf{v}^{t-h/2}}{2}. \tag{7.5}$$

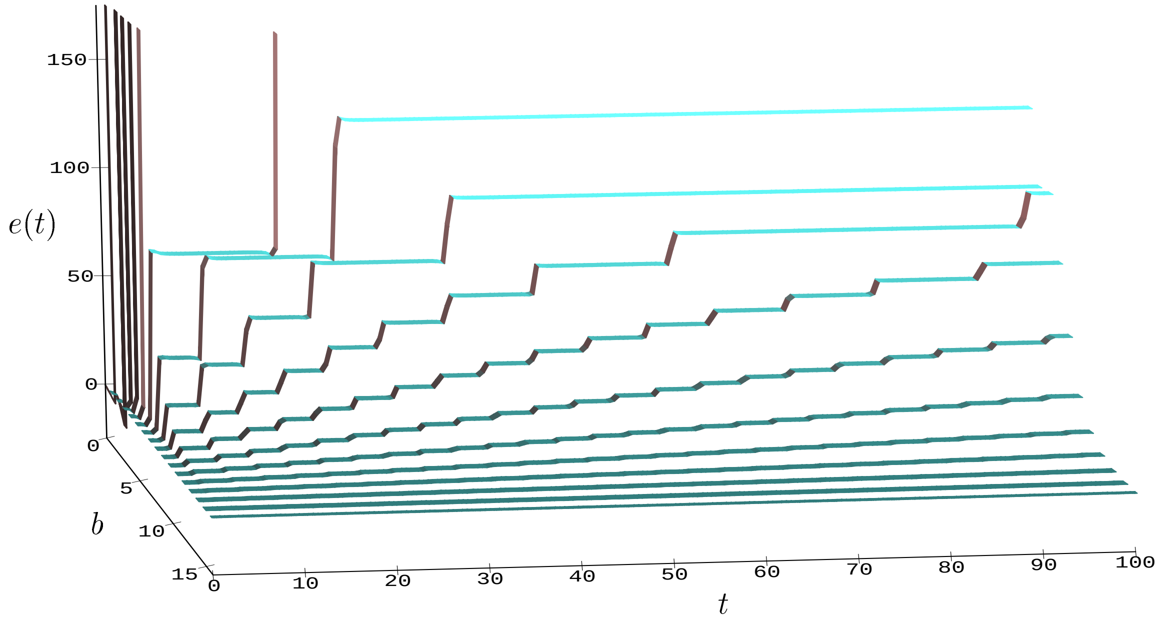
The same method can also be used to extrapolate the angular velocities.

Comparison

To provide a comparison between the various explicit integration schemes we prepared two tests which we executed for all three integration schemes over a variety of time steps. The first test is a simple two particle system without damping which allowed us to compare the stability and accuracy of integration. Without damping the particle system should conserve energy and thus we could measure the accuracy based on the relative error in the system energy. The second test used a complex system of 800 particles with both global velocity damping and relative inter-particle velocity damping. Since the energy in the system should decrease due to the damping terms, this test does not allow us to measure the accuracy using the relative error in system energy. However it does provide complex n-body interactions as we might expect in modeling and animation.



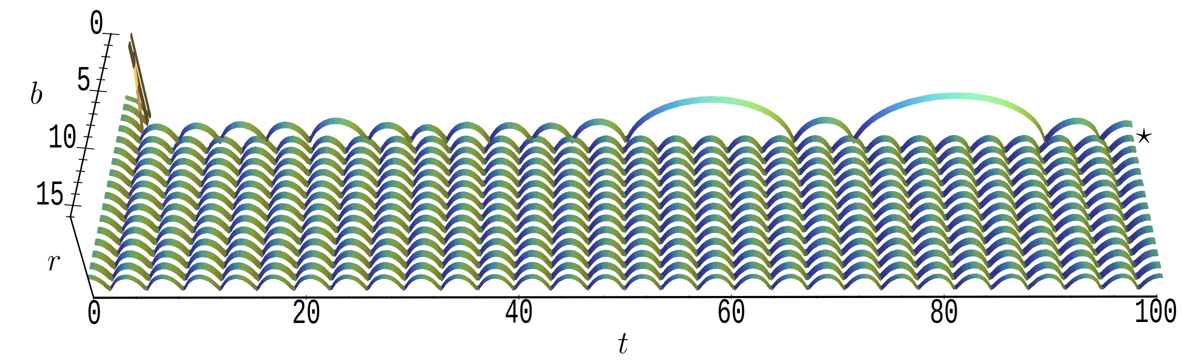
(a) Particle separation for a two particle system



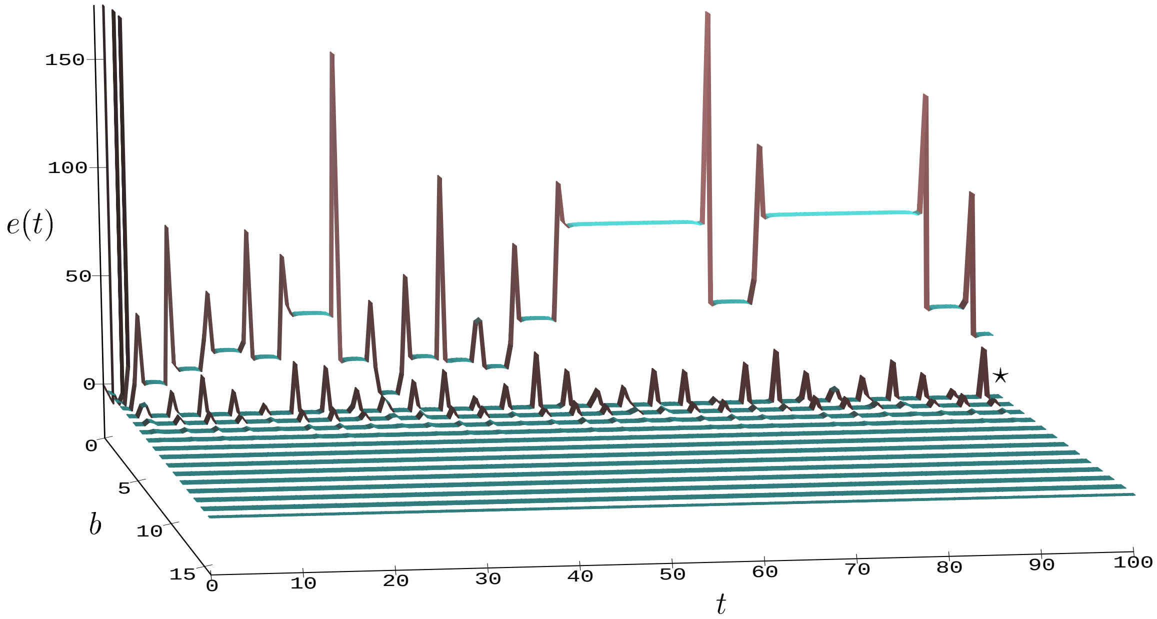
(b) Relative error of total system energy expressed as a percentage

Figure 7.2: Empirical results of modified Euler integration

The axes of the plots are as follows. Left to right is time t and ranges from 0 to 100. Back to front is logarithmic in the time step $h = 2^{-b}$ and ranges from 2^0 to 2^{-16}. (a) The vertical axis is particle separation $r = \|\mathbf{x}_1 - \mathbf{x}_2\|$ and ranges from 0 to 8. (b) The vertical axis is relative error e in system energy and ranges from -25% to 175%. For plotting, the data was reduced to 1601 samples for each simulation.



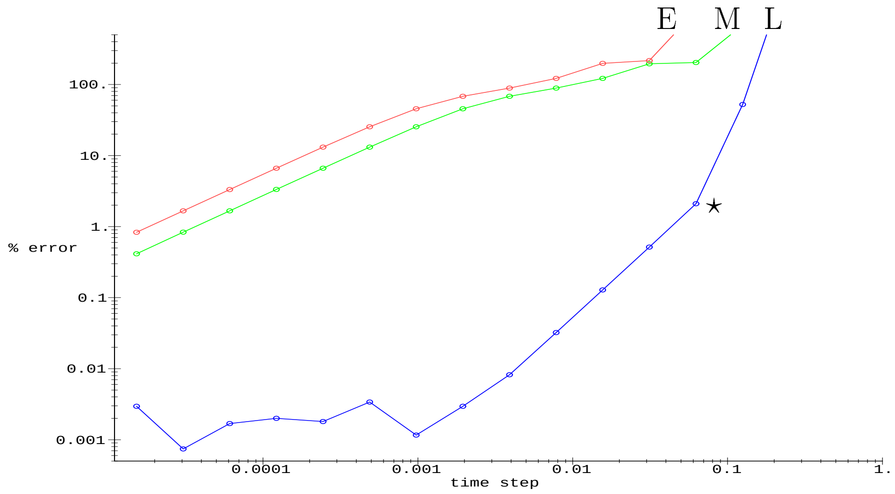
(a) Particle separation for a two particle system



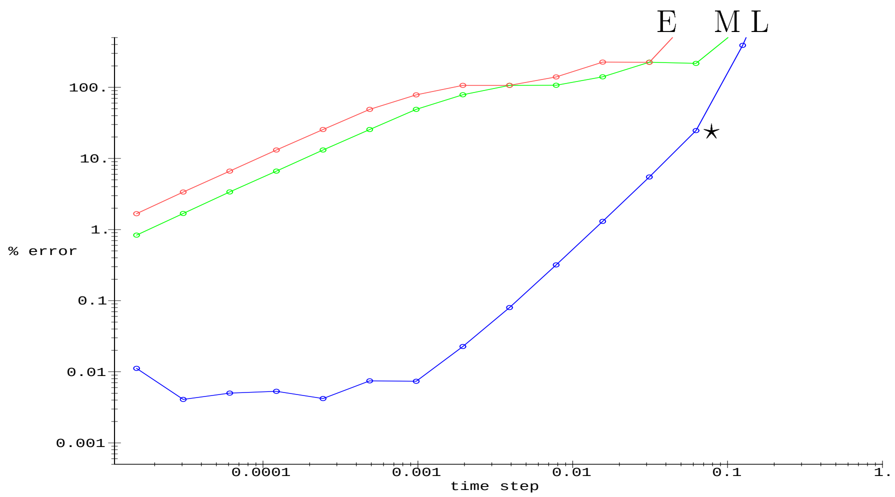
(b) Relative error of total system energy expressed as a percentage

Figure 7.3: Empirical results of Leapfrog integration

The axes of the plots are as follows. Left to right is time t and ranges from 0 to 100. Back to front is logarithmic in the time step $h = 2^{-b}$ and ranges from 2^0 to 2^{-16}. (a) The vertical axis is particle separation $r = \|\mathbf{x}_1 - \mathbf{x}_2\|$ and ranges from 0 to 8. (b) The vertical axis is relative error e in system energy and ranges from -25% to 175%. For plotting, the data was reduced to 1601 samples for each simulation.



(a) Average relative error in system energy



(b) Maximum relative error in system energy

Figure 7.4: Relative error in system energy

The horizontal axis is logarithmic in the time step h. (a) The vertical axis is logarithmic in the average relative system error over the simulation. (b) The vertical axis is logarithmic in the maximum relative system error over the simulation. The Euler method is displayed in red and labeled "E". The modified Euler method is displayed in green and labeled "M". The Leapfrog method is displayed in blue and labeled "L". The unusual results for the leapfrog for small relative error values ($e \approx 0.01\%$) we believe are due to limited machine precision in calculating the error measure. The $\star$ data points corresponds to the simulations with $\star$ symbols in Figure 7.3.

Comparison: First Test

For the first test the particle system studied was as follows:

- Particles interact according to the unweighted Lennard-Jones potential.
- Lennard-Jones parameters of $r_o = 1$, $n = 4$, $m = 2$, and $e = 1.0$.
- Two particles initially separated by 2 units.
- No neighborhood bounds, that is the particles are always coupled.
- No damping.

The system was integrated under the following conditions.

- For the Euler, modified Euler, Leapfrog explicit integration schemes.
- For times steps $h = 2^{-b}$ where $b = 0, 1, 2, \ldots, 16$.
- From time $t = 0$ to $t = 100$.

For each test case and for each time step executed we collected data on the particle separation, kinetic energy, and potential energy. The results from these simulations are displayed in Figures 7.2, 7.3, and 7.4.

In Figures 7.2(a) and 7.3(a) show the particle separation plotted as a height field for the modified Euler and Leapfrog schemes respectively. Time runs linearly, left to right. The time step is decreasing, back to front, on a logarithmic scale. The Euler method is not shown as it is not substantially different than the modified Euler, except for a factor of two as is explained shortly. Note that the modified Euler method is still converging to a solution for the smallest time step. This can be noticed by visually inspecting the right hand side of Figure 7.2 where differences in the period of oscillation can be seen.

Figures 7.2(b), and 7.3(b) display the relative error in the total system energy. The relative error e is computed as the percentage difference in system energy at time t to original system energy

$$e(t) = \frac{E_S(t) - E_S(0)}{E_S(0)} \times 100$$

where $E_S(0)$ is the sum of kinetic and potential energies at time 0

$$\begin{aligned} E_K(0) &= 0 \\ E_P(0) &= \phi_{\mathrm{LJ}}(2) = -0.4375 \end{aligned}$$

and $E_S(t)$ is the sum of kinetic and potential energies at time t

$$\begin{aligned} E_K(t) &= \frac{1}{2}\left(m_1\|\mathbf{v}_1(t)\|^2 + m_2\|\mathbf{v}_2(t)\|^2\right) \\ E_P(t) &= \phi_{\mathrm{LJ}}(\|\mathbf{x}_1(t) - \mathbf{x}_2(t)\|). \end{aligned}$$

Figure 7.4(a) displays the average value of the relative error over the entire run of each simulation. Figure 7.4(b) displays the maximum value of the relative error over the entire run of each simulation. For both cases the absolute value of the relative error is used.

The differences between the Euler method and the modified Euler is as expected. Analysis using a Taylor series expansion shows that the second step of the modified Euler method has half the local truncation error than does the second step of the Euler method. This is noticeable in the results. The data shows the modified Euler with a time step of $2h$ produces results almost identical to the Euler method with time step of h. Both Euler methods have a local truncation error of $O(h^2)$ per step.

The Leapfrog method performed well in practiced. As the time step decreased, it quickly converged to a stable solution. Analysis using a Taylor series expansion shows Leapfrog exhibits a local truncation error of $O(h^3)$ per step and thus global truncation error of $O(h^2)$. The rate of decrease of error in the data, matches this analysis.

When viewing the raw data for the various integration schemes, at stable time steps, the Euler methods almost exclusively exhibited positive relative error, though over small intervals it exhibited negative relative error. The Leapfrog exhibited both positive and negative relative errors in nearly equal amounts. We speculate the difference is due to the nature of the potential energy function combined with the fact that the Leapfrog is a time centered scheme, while the Euler methods are not. However, the exact reason for these difference requires further study.

Comparison: Second test

For the second test the particle system studied was as follows:

- A particle system containing 800 particles.
- Particles interacting according to the weighted Lennard-Jones potential, global velocity damping, viscous inter-particle damping, and limited particle interaction.
- The parameters of the Lennard-Jones potential were $r_o = 1$, $n = 4$, $m = 2$, and $e = 1$.
- The parameters of the weighting function were $r_a = r_o$ and $r_b = 1.7r_o$.
- Neighborhood range was set to $1.7r_o$.
- Neighbors were recomputed after every time step.
- The global velocity based damping was $-0.25\dot{\mathbf{x}}_i$.
- The inter-particle viscous damping was $-0.125\,(\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j)$.
- In the case of the Leapfrog integration scheme, extrapolated velocities defined by (7.5) were used in the damping computations.

We ran the test on a particle system of 800 particles that was initially positioned on a hexagonal grid of $8 \times 10 \times 10$ with particles separated by $1.5r_o$. The separation value was picked to create a large initial system energy. To provide for more complex behavior, the positions were slightly displaced by random amounts. This removed the strong spatial symmetry of the initial placement. The displacement was computed as a vector of three independent observations of a Gaussian random variable with zero mean and a variance of 0.05. If the magnitude of the displacement was greater than the variance, then the displacement vector was scaled to be of length equal to the variance. This was to keep particles receiving a large random displacement from deviating too far from their initial position.

The system was integrated under the following conditions:

- For the Euler, modified Euler, Leapfrog explicit integration schemes.
- For times steps $h = 2^{-b}$ where $b = 3, 4, 5, 6, 7$. This is equivalent to the range $h = 0.125$ to $h = 0.0078125$.
- From time $t = 0$ to $t = 20$.

The initial configuration for the tests is shown in Figure 7.5. The particles are color coded based on the number of interacting neighbors. Particles on the edge boundaries have fewer neighbors and are darker in color. Some particles on the face have more neighbors than others due to the initial random displacements. The lines drawn between particles are the neighbor connections.

For each test case and for each time step executed, we collected data on the system kinetic energy and system potential energy. The results from these simulations are displayed in Figures 7.6, 7.7, 7.8, and 7.9. The system energies are plotted in the left column and are labeled by (a), (c), and (e). Images from the final frame of each animation are displayed next to the energy plots and are labeled by (b), (d), and (f). The results for each time step are shown on the same page. The results are presented in the order of increasing time step size.

The first set of results is for the time step of $\Delta t = 2^{-7}$ (shown in Figure 7.6) and show all three integration schemes as being stable. The initial cube of particles collapses into a ball as the system energy is minimized. The plots of energy show steadily decreasing potential energy (the lower line of each plot) indicating this is a highly damped system. The difference between the total system energy (top line of each plot) and the potential energy is the amount of kinetic energy in the system. The negative total potential energy (and hence system energy) is due to the definition of the Lennard-Jones potential. For the same time step in the two-particle undamped system, the Euler method was unstable. The inclusion of damping has stabilized the Euler method in this test, even though the interactions are more complex.

The second set of results (shown in Figure 7.7) is for the same initial configuration, yet the a time step was twice as large. Both the Leapfrog and modified Euler method produce similar stable results, but the Euler method exhibits instabilities. These are seen as large spikes in the system energy plot. The large potential energy values indicate that particles came closer than the collision distance. Finally, the flattening

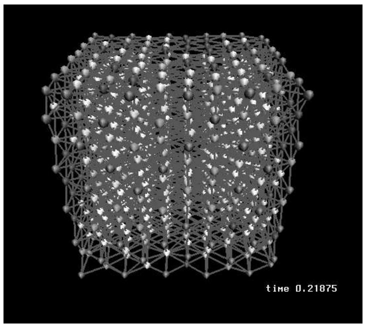


Figure 7.5: Initial particle system configuration

This is the initial particle system configuration of 800 particles arranged on an hexagonal grid with random displacements assigned to each particle.

---

of the potential energy curve to near zero energy indicates that only a few of the particles are interacting with nearby particles.

The third set of results (shown in Figure 7.8) is for time step twice as large again. Both Euler and modified Euler methods exhibit instabilities, with the Euler method becoming unstable almost immediately. The modified Euler method became unstable at about the same time that the Euler method became unstable in the previous experiment. The Leapfrog method remained stable.

The fourth set of results (shown in Figure 7.9) is for a time step twice as large again. In this case all three integration methods became unstable. In the two particle system, the Leapfrog scheme was stable at this time step. Since the interactions in this system are much more complex, this is not a surprising result.

In summary, although the Euler and modified Euler method performed better with the inclusion of damping, the Leapfrog stability limit was still four times that of the Euler method, and twice that of the modified Euler method.

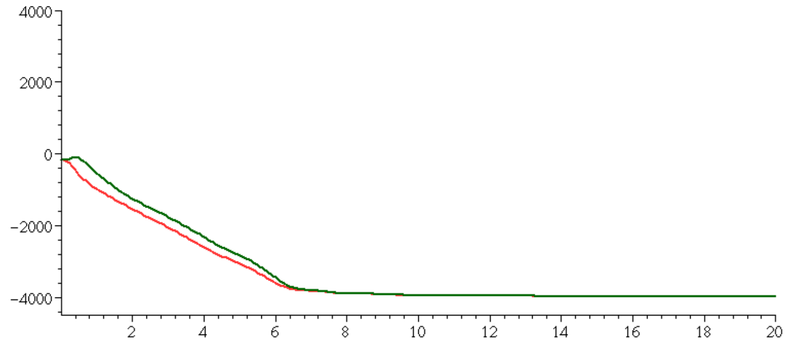
Additional measures

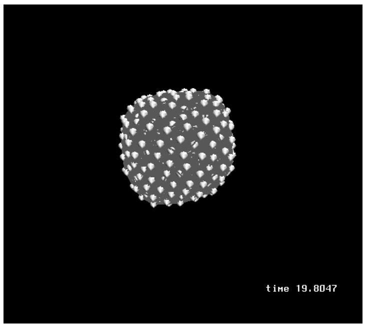
To allow us to take larger time steps, then would be otherwise be allowable, we can apply the following measures.

- We can introduce limits on the maximum velocity and maximum change in velocity.
- We can place limits on the inter-particle forces, such as clipping the magnitude of Lennard-Jones force for distances less than the collision distance to the value at the collision distance.

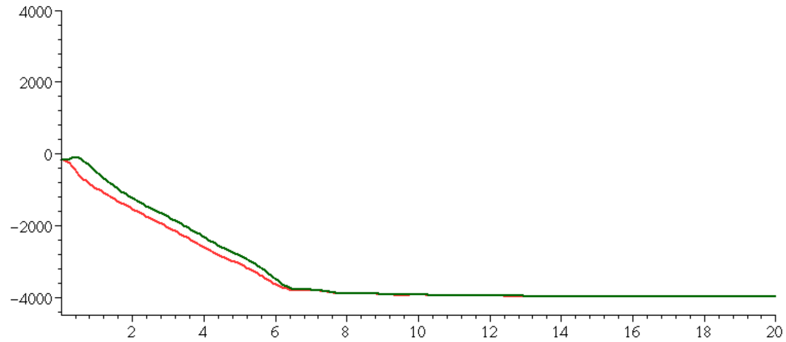
These measures in effect reduce the natural frequencies of the system resulting in stable integration at larger time steps.

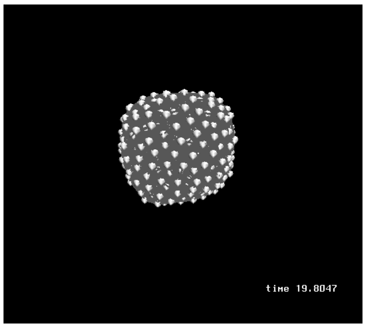
From our testing it became obvious that the majority of error for the Euler method was introduced when integrating over regions with large changes in force. Since the



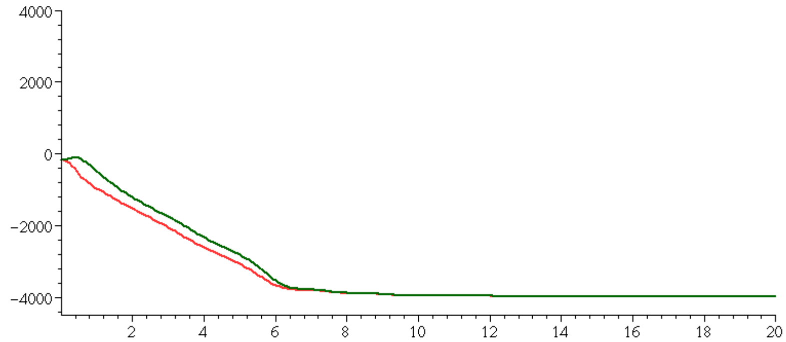


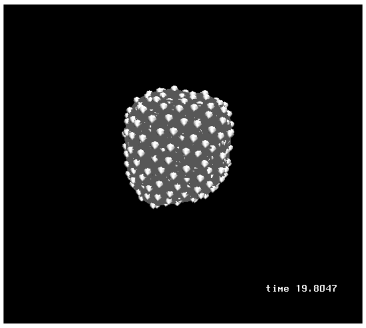
Leapfrog (a) (b)





Modified Euler (c) (d)

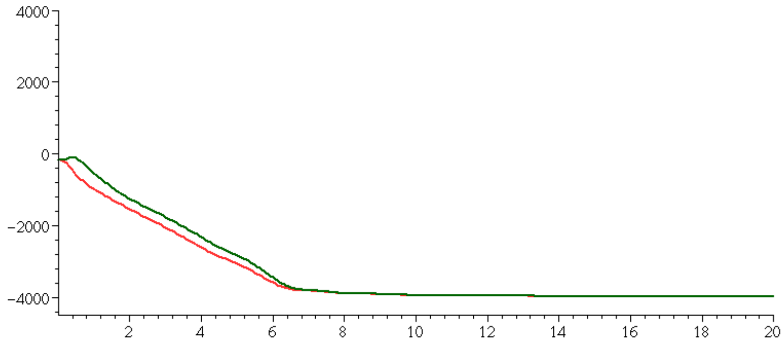


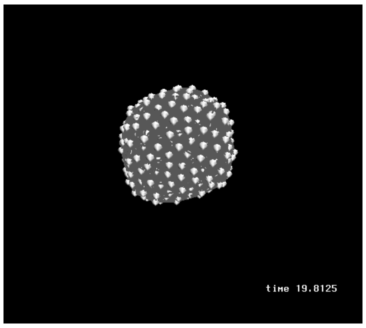


Euler (e) (f)

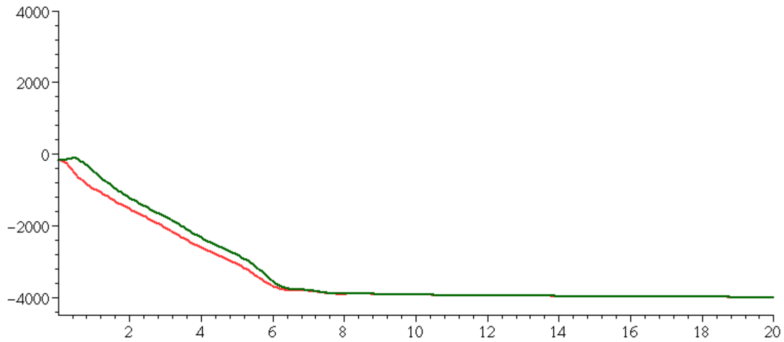
Figure 7.6: Integration with $\Delta t = 2^{-7}$.

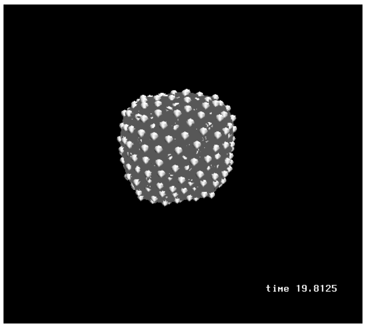
The rows above display the results for Leapfrog, Modified Euler, and Euler schemes respectively. Left column lists the plots of system energy versus time. The right column displays the final image from each animation. The dark line (top/green) is the total system energy $E_S = E_P + E_K$ and the lighter line (bottom/red) is the system potential energy E_P. Note $E_S(t) \geq E_P(t)$. For this time step, all the integration schemes were stable.



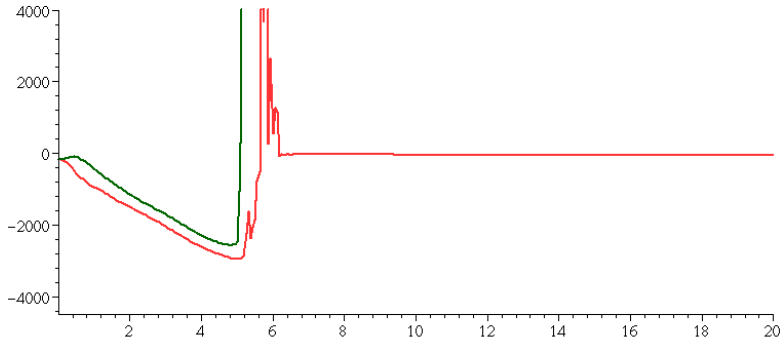


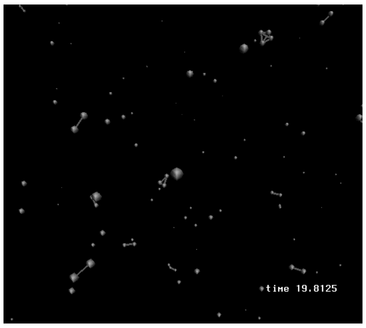
Leapfrog (a) (b)





Modified Euler (c) (d)





Euler (e) (f)

Figure 7.7: Integration with $\Delta t = 2^{-6}$.

The rows above display the results for Leapfrog, Modified Euler, and Euler schemes respectively. Left column lists the plots of system energy versus time. The right column displays the final image from each animation. The dark line (green) is the total system energy $E_S = E_P + E_K$ and the light line (red) is the system potential energy E_P. Note $E_S(t) \geq E_P(t)$. The spikes in the bottom energy plot indicate instabilities in integration. The result was that the particle system exploded.

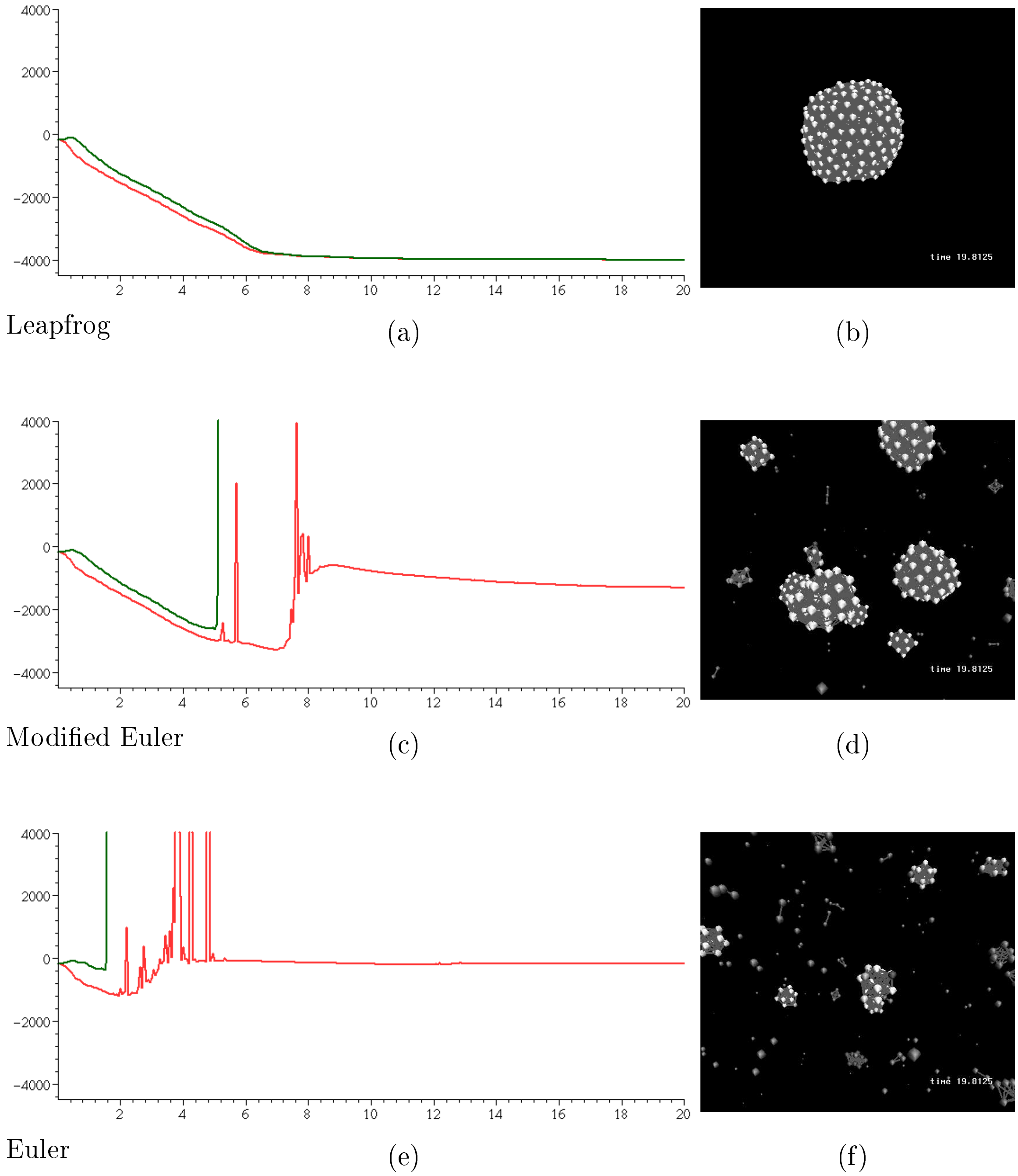
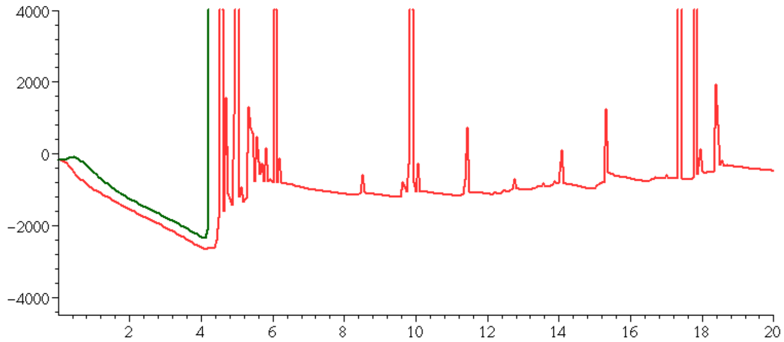
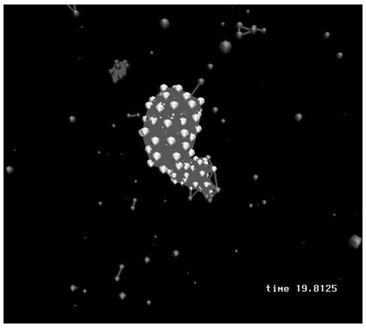


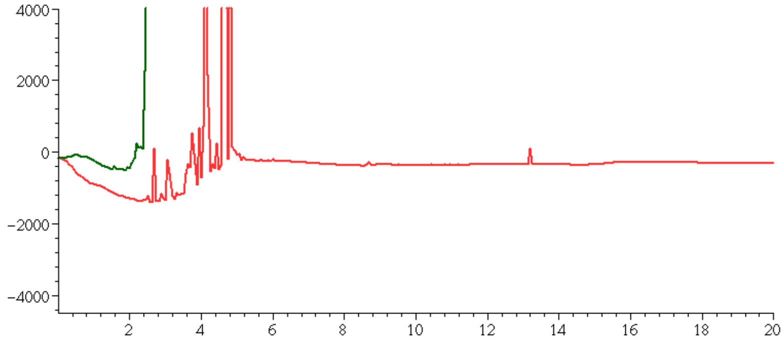
Figure 7.8: Integration with $\Delta t = 2^{-5}$.

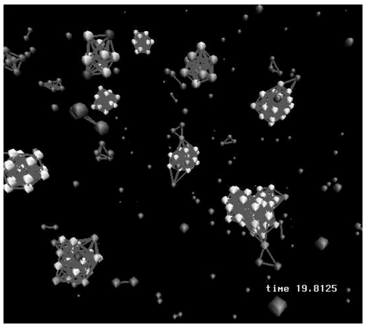
The rows above display the results for Leapfrog, Modified Euler, and Euler schemes respectively. Left column lists the plots of system energy versus time. The right column displays the final image from each animation. The dark line (green) is the total system energy $E_S = E_P + E_K$ and the light line (red) is the system potential energy E_P. Note $E_S(t) \geq E_P(t)$.



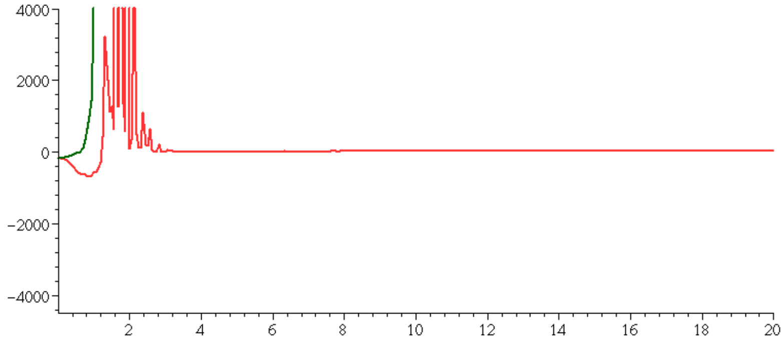


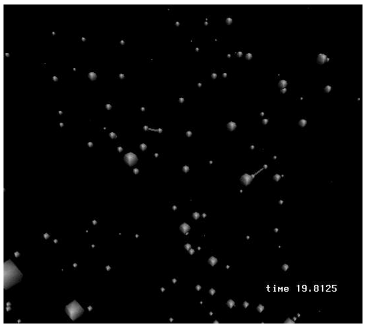
Leapfrog (a) (b)





Modified Euler (c) (d)





Euler (e) (f)

Figure 7.9: Integration with $\Delta t = 2^{-4}$.

The rows above display the results for Leapfrog, Modified Euler, and Euler schemes respectively. Left column lists the plots of system energy versus time. The right column displays the final image from each animation. The dark line (green) is the total system energy $E_S = E_P + E_K$ and the light line (red) is the system potential energy E_P. Note $E_S(t) \geq E_P(t)$.

repulsive force between particles, due to the Lennard-Jones potential, is stronger than the attractive forces, more error tends to be introduced when particles are colliding than when they are moving apart. This can be seen in the graph of the relative error for the Euler method (Figure 7.2(b)) as "steps" of increasing error. It can also be seen as spikes in the error plot (Figure 7.3(b)) of the Leapfrog method. The net result is that the system energy increases and the system eventually explodes as the accumulated error exceeds the binding (potential) energy of the system. For modest time steps, the introduction of damping overcomes the accumulation of error, reversing the process. From experience, both global velocity based damping and viscous damping units are effective (Section 3.3.2). The viscous damping unit is superior for stabilizing numerical errors because it is based on the relative velocity between particles and is independent of rigid body motion.

7.3.4 Stability and Accuracy Analysis

Instead of empirically testing for a suitable time step, we can analytically derive stability and accuracy criteria for the leapfrog integration scheme. For simplicity we analyze a two particle system interacting according to the Lennard-Jones potential. This provides a starting point for deriving the stability criteria of more complex systems and thus choosing an appropriate time step.

Stability

Solving for error propagation, (Hockney and Eastwood, 1988, pp. 97–106) derives the stability boundary

$$\omega \Delta t < 2, \tag{7.6}$$

where ω is the highest oscillation frequency of the system: for the second-order differential equation of motion for undamped motion between a pair of particles, when approximated by the time-centered leapfrog scheme. There are two basic assumptions made in deriving the boundary: (1) the solution of the differential equation is oscillatory in nature and (2) the solution is derived for the worst case. The worst case is defined by the highest frequency ω of the system, which occurs at the maximum change in the magnitude of the force

$$\omega^2 = -\frac{1}{m}\frac{\mathrm{d}f}{\mathrm{d}x} \tag{7.7}$$

for negative forces.

The Frequency of Oscillation

For a general force $f(x)$, equations (7.6) and (7.7) allow us to relate the force to the highest oscillation frequency of the system and from that determine the stability and accuracy for a given time step. We determine the time step for a system of particles absent of any damping forces and interacting according to the Lennard-Jones potential energy function. We choose the Lennard-Jones potential because of all our inter-particle potentials it presents the largest magnitude in change of force.

We follow a one-dimensional argument, similar to that given in (Hockney and Eastwood, 1988, pp 461-465). We begin by estimating the highest frequency which occurs when the maximum gradient of the force is present. For the case of the Lennard-Jones potential, this occurs when the neighboring particles move towards each other to a position of closest approach. The equations of undamped motion for two such particles are

$$m_1 \frac{d^2 x_1}{dt^2} = -f(x_2 - x_1) \tag{7.8}$$

$$m_2 \frac{d^2 x_2}{dt^2} = f(x_2 - x_1), \tag{7.9}$$

where f is the magnitude of the repulsive force, x_1 and x_2 are the positions of the two particles, and m_1 and m_2 are the respective masses. We choose to analyze the repulsive force of the Lennard-Jones function, over the attractive force, because the repulsive force exhibits higher values of magnitude. Multiplying the first equation by m_2 and the second equation by m_1, and subtracting yields

$$m_1 m_2 \left(\frac{d^2 x_2}{dt^2} - \frac{d^2 x_1}{dt^2} \right) = m_1 f - m_2 f. \tag{7.10}$$

Reducing, we get the equation of relative motion

$$m^\star \frac{d^2 r}{dt^2} = f(r), \tag{7.11}$$

where $r = x_2 - x_1$ is the particle separation, and $m^\star = m_1 m_2/(m_1 + m_2)$ is the reduced mass. Since f(r) is non-linear, to obtain an effective frequency we consider a small perturbation substituting $r + r'$ for r, where r' is small

$$m^\star \frac{d^2 (r + r')}{dt^2} = f(r + r'). \tag{7.12}$$

We expand the right side as a Taylor series, keeping only the first two terms of the series.

$$f(r + r') \simeq f(r) + r' \frac{\mathrm{d}f}{\mathrm{d}r} + \dots$$

The left side expands to

$$m^\star \frac{d^2 (r + r')}{dt^2} = m^\star \left(\frac{d^2 r}{dt^2} + \frac{d^2 r'}{dt^2} \right).$$

Expanding (7.12) and subtracting (7.11) gives

$$m^\star \frac{d^2 r'}{dt^2} = r' \frac{\mathrm{d}f}{\mathrm{d}r}. \tag{7.13}$$

Rewriting results in the linearized equation for the perturbation r'

$$\frac{d^2 r'}{dt^2} = -\omega^2 r', \tag{7.14}$$

where

$$\omega = \left(-\frac{1}{m^\star}\frac{\mathrm{d}f}{\mathrm{d}r}\right)^{\frac{1}{2}} \tag{7.15}$$

according to equation (7.7).

When df/dr is negative, as it is for the short-range Lennard-Jones repulsion, equation (7.14) is the differential equation for simple harmonic oscillation with frequency ω given by (7.15). From (7.7) we can see that the maximum frequency will occur for small masses and steeper laws of repulsion between the particles. For the case of the Lennard-Jones potential, the frequency is given by

$$\omega = \left(-\frac{1}{m^\star}\left(\frac{B^\star}{r^{n+2}} - \frac{A^\star}{r^{m+2}}\right)\right)^{\frac{1}{2}}, \tag{7.16}$$

where $A^\star$ and $B^\star$ are the constants

$$A^\star = \frac{\epsilon nm(m+1){r_o}^m}{n-m} \qquad B^\star = \frac{\epsilon nm(n+1){r_o}^n}{n-m}. \tag{7.17}$$

This expression for ω indicates that as the minimum separation r goes to zero, the frequency of oscillation goes to infinity. Since stability requires $\omega\Delta t < 2$, all time steps, in principle, are unstable. However let us assume that the particles do not come closer than a given distance. For the purpose of this analysis we will assume this distance to be the collision distance σ of the Lennard-Jones potential. In order to prevent the few particles that might end up closer, from becoming unstable, one can limit the force of repulsion for distances less than the collision distance to its value at the collision distance.

We now compute the oscillation frequency for two cases: a worst case at the collision distance, and an average case at the equilibrium spacing. We let the defining parameters of the Lennard-Jones functions be $m = 2$, $n = 4$, $\epsilon = 1$, and $r_o = 1$. And we let the mass for each particle be 1.0 so $m^\star = 0.5$. This results in a frequency for the worst case of 15.0 and for the average case of 4.0.

Accuracy

We now consider the issue of accuracy. Let's assume particles will have an oscillation frequency ω associated with the average separation of r_o. If the leapfrog scheme is used to integrate the equations of motion, then (7.14) is approximated as

$$\frac{r'(t+\Delta t) - 2r'(t) + r'(t-\Delta t)}{(\Delta t)^2} = -\omega^2 r'(t) + \frac{(\Delta t)^2}{12}\frac{d^4r'}{dt^4} \tag{7.18}$$

based on the Taylor series. For an oscillation at frequency ω, we apply equation (7.14) to find

$$\frac{d^4r'}{dt^4} = \omega^4 r'. \tag{7.19}$$

The ratio of truncation error (last term on the right-hand side of (7.18)) to the true force (first term on the right-hand side of (7.18)) is

$$\frac{(\omega\Delta t)^2}{12}. \tag{7.20}$$

Based on the above error measure, we again consider two cases: an average case and a worst case. Recall that from equation (7.6), the product of the frequency and time step should be less than 2 for stability. If we let the product equal 0.25 as a condition for sufficient integration for average separation, by (7.15) we can compute ω, and the relative truncation error from (7.20) reduces to 1/2 of a percent. This appears to be of reasonable accuracy for the average case. In the worst case, we assume that few particles will be at the collision limit. Thus we favor a larger time step over increased accuracy. By choosing a product of 1, we get a relative truncation error of 8 percent. A time step satisfying both accuracy and highest frequency constraints is the minimum time step of the two computations.

Having computed the worst case and average case frequencies of the Lennard-Jones function earlier, we now compute time steps. The conditions on the time step can be summarized as follows

- Accuracy at the average separation:

  $r = r_o = 1, \qquad \omega = 4, \qquad \omega\Delta t = 0.25, \qquad \text{error} = 0.5\%, \qquad \Delta t = 0.0625$

- Stability at the highest frequency:

  $r = \sigma = 1/\sqrt{2}, \qquad \omega = 15, \qquad \omega\Delta t = 1.0, \qquad \text{error} = 8\%, \qquad \Delta t = 0.0667$

for the Lennard-Jones parameters $m = 2$, $n = 4$, and $\epsilon = 1$. Our analysis matches the empirical results of Figures 7.2, 7.3, and 7.4. In Figures 7.3 and 7.4 the time step of 0.0625 is marked with a $\star$ symbol. For this time step, the average error was 2.1% over the entire simulation and the maximum error for any step was 24.7%.

7.3.5 Timing Results

To measure the speed at which one can currently calculate a simulation, we ran several tests. Rather than quoting the speed in frames per second, we found it more useful to quote the results in particle pair computations per second. Given a rate of pairs per second, the frame rate for most scenes can be approximated by estimating the number of time steps taken per rendered frame and the number of neighbors per particle. The number of neighbors per particle is dependent both on the neighborhood range and whether the model is using volume or surface particles. The system we tested consisted of 1000 volume particles interacting with their nearest neighbors. The weighting function parameters were $r_a = r_o$ and $r_b = 1.4r_o$, and the neighborhood range was $1.4r_o$. Neighbors were computed every 10 steps. The particles interacted according to the Lennard-Jones potential, inter-particle viscous damping, and global velocity damping. Approximately 19,000 particle pair interactions were computed per second when executing on an SGI O2. That amounts to 6000 pairs per time step, or 3 steps per second [7]. Using the same number of oriented particles would result in approximately 4 to 6 steps per second, depending on the number of surface potentials used. It should be noted that the code executed in this test was not optimized, but rather written for generality. From prior experience, profiling and optimizing the code should give a speed up factor on the order of 4 to 8 times the unoptimized code speed. Such optimizations would result in 12 and 16 steps per second on the low end, and 24 and 48 steps per second on the high end for particle systems of 1000 volume and surface particles respectively.

7.3.6 Discussion

The equations of motion for a dynamically coupled particle system result in two sets of coupled second order non-linear differential equations (3.1) and (4.1). To determine the position and orientations of the particles at new time values, we need to solve the initial value problem for these equations. To do this, we rewrite the second order system of differential equations as sets of coupled first order differential equations and then numerically integrate the first order equations through time using a finite difference based scheme.

Implicit or explicit integration schemes can be used to integrate over a given time step. When the system dependencies can be written in matrix form, the matrix will be large, unstructured, and sparse. In addition, the non-zero values of the matrix will be continually changing making optimizations more difficult. High order explicit schemes, such as Runge-Kutta (Press et al., 1988) or semi-implicit methods (Terzopoulos et al., 1987) could be used, and for typical dynamic systems would result in good convergence and large time steps, but at the expense of a complicated implementation and possibly slow interactive response. When considering these alternatives we must keep in mind that the time step will be limited by the natural frequency of the system and that the system in question is highly correlated and oscillatory in nature. For the case of a damped non-linear oscillator, the explicit leap-frog scheme, with stable time step, is more economical than the implicit Euler scheme while yielding almost identical results (Greenspan, 1973). For most n-body calculations, simple second-order accurate explicit schemes, such as Leapfrog, provide the best compromise between accuracy, stability, and efficiency (Hockney and Eastwood, 1988). To maintain interactive update rates and simplicity of implementation, we have used both the explicit Euler and Leapfrog integration schemes. Empirical results correspond with analytical derivations showing the Leapfrog method to be superior.

[7] The exact results were 5940 pairs per step, 18,978 pairs per second, and 3.19 steps per second.

7.4 Visualization

7.4.1 Rendering

Rendering, as we know from the ray-tracing and radiosity literature, can be very time consuming. In general, high quality rendering is based on determining the reflection of light off of a continuous surface with associated material attributes. In Chapter 5 we discussed the generation of continuous surface descriptions and the rendering of those surfaces.

However we need not assume that a continuous description surface is necessary for high quality renderings. There are many cases when a surface description is not necessary or desirable when rendering an object's shape. If the object being modeled is gas-like or of an amorphous nature without definite boundaries, then the particle system can be rendered directly, thus bypassing the surface description phase. For example, Reeves (1983b) and Sims (1990) have rendered particle systems as *light emitting points* to simulate fire, water falls, and mist. Alternatively if one has assigned a *density field* to each particle, one can integrate over the density fields as Stam (1995) has done to create visually complex scenes of clouds, smoke, and fire. For rendering similar types of fluid-like behavior, 4D texture maps could also be used to interesting effect. Since the positions of an oriented particle model provide a uniform surface sampling of the object, particle systems lend themselves very well to rendering with *paint strokes* (Meier, 1996) resulting in a "painterly" rendering style.

7.4.2 Visual Cues for Real-Time Use

For interactive modeling, we prefer rendering techniques which maximize our understanding of the system. Except for the final stages of an animation, it makes more sense to spend the computing power on providing informative displays for the user instead of regenerating complete surface descriptions. The display should convey information we are interested in such as particle position, orientation, energy, neighbor connections, and provide quick approximations of the final surface. Such real-time visual cues are indispensable in debugging, scripting animations, and modeling. Here we review some techniques we have found useful.

Light Emitting Points

Rendering each particle as light emitting points is the simplest and quickest method. The result is an uncluttered scene in which all of the particles are visible. X-Y position information is obvious and depth perception is enhanced by rotating the scene in real-time. Additional information can be encoded in the color of the particle. For example the heat energy of a particle is easily displayed as a variation from white (cold) to red (hot).

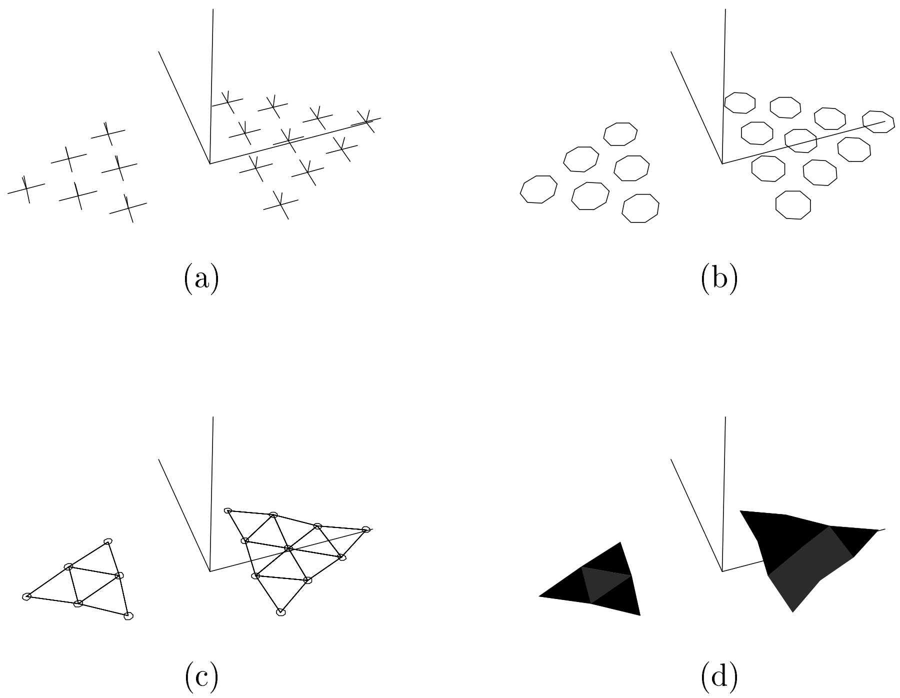


Figure 7.10: Visualization techniques

(a) axes, (b) discs, (c) wireframe triangulation (d) flat-shaded triangulation.

Wire Frame Particles

Wire frame representations of a particle are useful for approximating the volume or surface area represented by a particle without obscuring the other particles. For a solid modeling particle system, we draw the particle as a star: three mutually perpendicular line segments. Oriented particles can be drawn with the positive Z axis highlighted to indicate orientation. We have found displaying a wire frame hexagon for each oriented particle results in an intuitive feel of the surface while still conveying orientations (Figure 7.10).

Surface Approximations

To quickly approximate the surface of a solid model, the particles can be displayed as spheres or cubes. Another simple approximation of the iso-surface is to render a sphere for each particle and shade the sphere according to the gradient of the summed fields at that point (Miller and Pearce, 1989). To approximate the surface resulting from oriented particles, the particles can be displayed as filled hexagons. Displaying the neighbor connections between particles as line segments is another method for quickly approximating the surface while at the same time conveying the structure of the particle system. A slightly longer process is to display a wire triangulation of the surface (Figure 7.10). The triangulation process is given in Section 5.2.3.

7.5 Summary

Our dynamically coupled particle system presents several computational challenges. In particular, the continually changing spatial relationships between particles complicates a variety of tasks. The computation of inter-particle forces, particle creation heuristics, and the triangulation algorithm are defined over all particle pairs making these $O(N^2)$ problems that must be solved at each time step. Our solution is to convert the problems from global problems involving all N particles, to local problems involving at most a constant number of nearby particles. This effectively transforms the problem to $O(N)$ time plus the time needed to find the neighboring particles.

The nearby neighbors problem, or the range search problem, is the problem of finding all particles within a given distance from a point in space. It computes the spatial relationships between points, suggesting we should organize the data according to the embedding space of the points. By using spatial subdivision techniques, which allow one to focus on relevant subsets of data, the range search problem can be solved on average in time $O(N)$ time for volume modeling using a fixed grid, and $O(N \log N)$ time for surface modeling using a region-based tree structure. To reduce computation during a simulation, we cache the results of neighbor computations for several time steps.

In order to compute the state of the system at new time values, we first calculate the inter-particle and external forces. To compute the forces in linear time, we limit the inter-particle force functions to a fixed distance bound, ignoring distant particles. We use a weighting function to decay our potential energy fields to zero at a fixed distance. In addition to reducing force computations, this approach provides a correct energy based representation of the splitting and merging of objects.

The differential equations of motion describe how a system of particles will respond to inter-particle and external forces. Equations (3.1) and (4.1) are second order nonlinear differential equations, making them difficult to integrate analytically. Instead, numerical integration techniques, based on a Taylor series expansion, are used to approximate the solution by taking finite steps through time. To do this, we reduce the second order set of differential equations of motion to two first order equations and solve the initial value problem forward in time. We have used simple explicit integration methods rather than more complicated implicit methods. We have empirically compared two low order explicit schemes. We also analyzed the leap frog method to derive analytic equations of stability and accuracy. Our empirical results match well with theoretical analysis. More sophisticated numerical high order explicit schemes, semi-implicit, and implicit schemes could also be used but it is unclear whether they would be fast enough for interactive applications. Physicists studying n-body problems have asserted that simple 2nd order time-centered explicit schemes work as well or better than low order implicit schemes, thus supporting our decision.

For interactive rendering of state we have suggested a variety of rendering styles (Section 7.4). A discussion of generating continuous surface descriptions is given in Chapter 5.

Through spatial subdivision of space and the use of explicit integration schemes, we have been able to reduce the total computational complexity of our system to expected computation times of (N) for volumes and $O(N \log N)$ for surfaces.

Chapter 8

Applications

8.1 Computer Assisted Animation

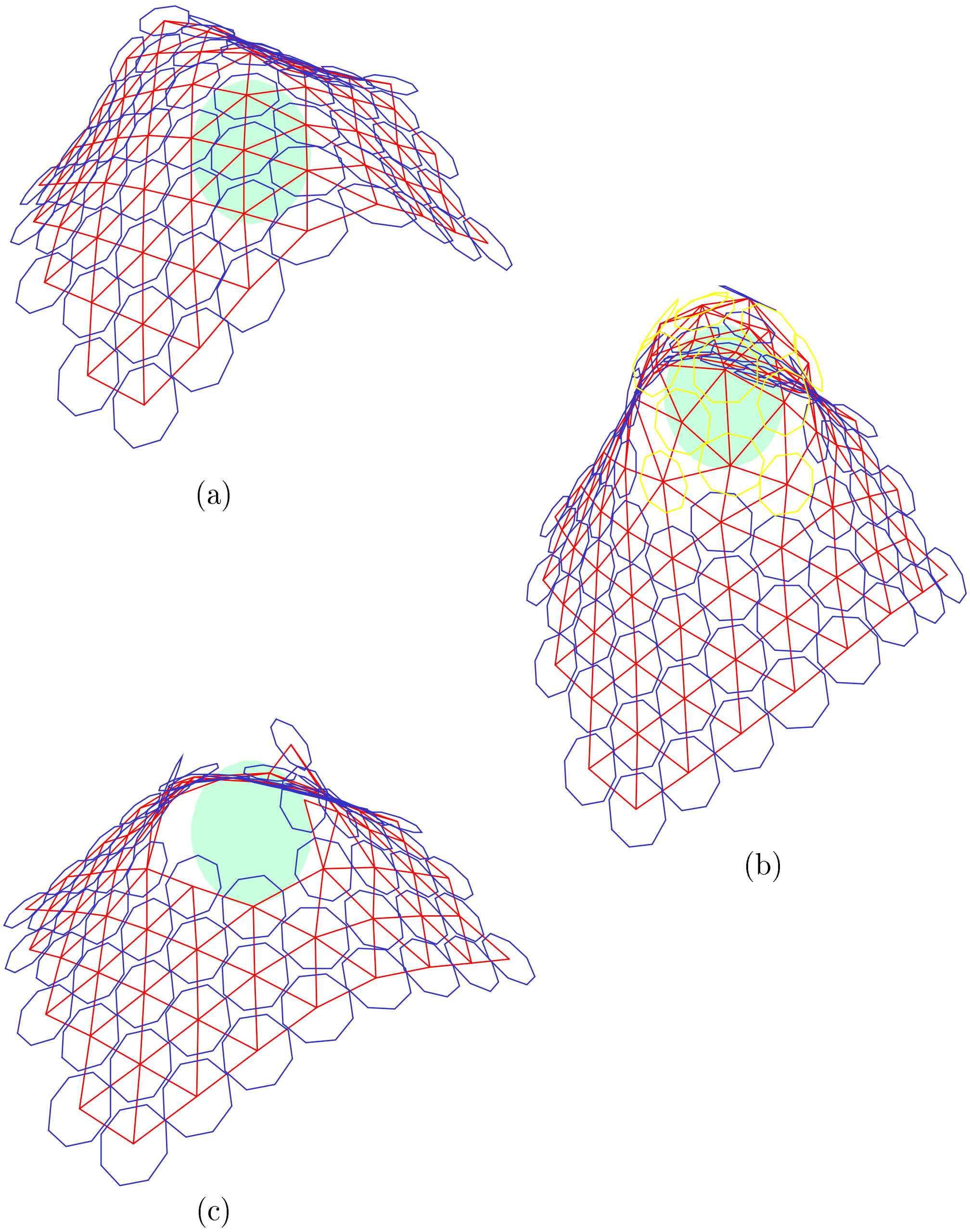
For computer assisted animation, we desire a model that can provide a wide range of dynamic properties and the ability to interact with an environment. Our particle systems can simulate a range of behavior in both the volume and surface models. We simulate solid to fluid behavior, infinitely stretchable 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



(a) (b) (c)

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.

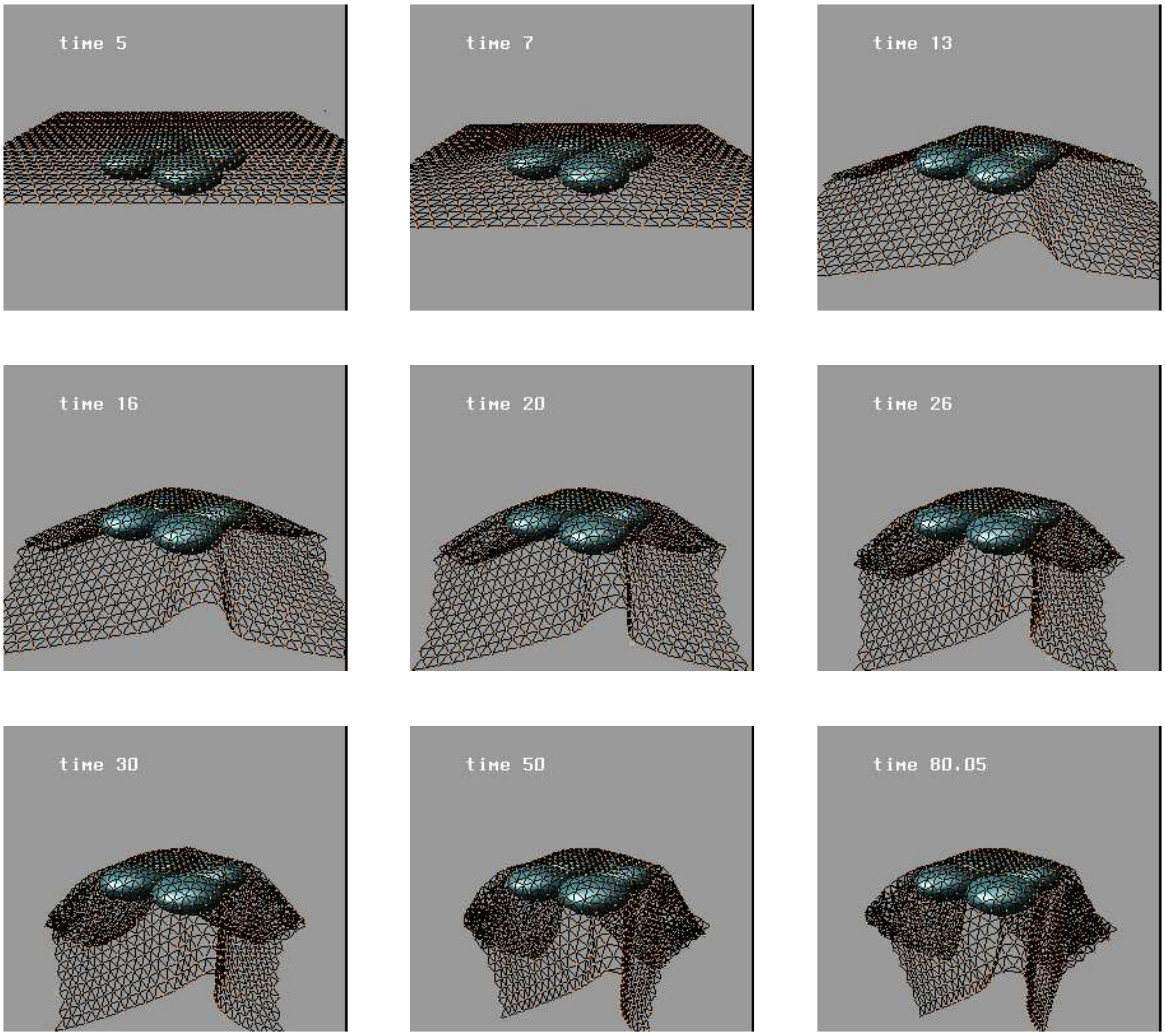


Figure 8.2: Draping cloth

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.

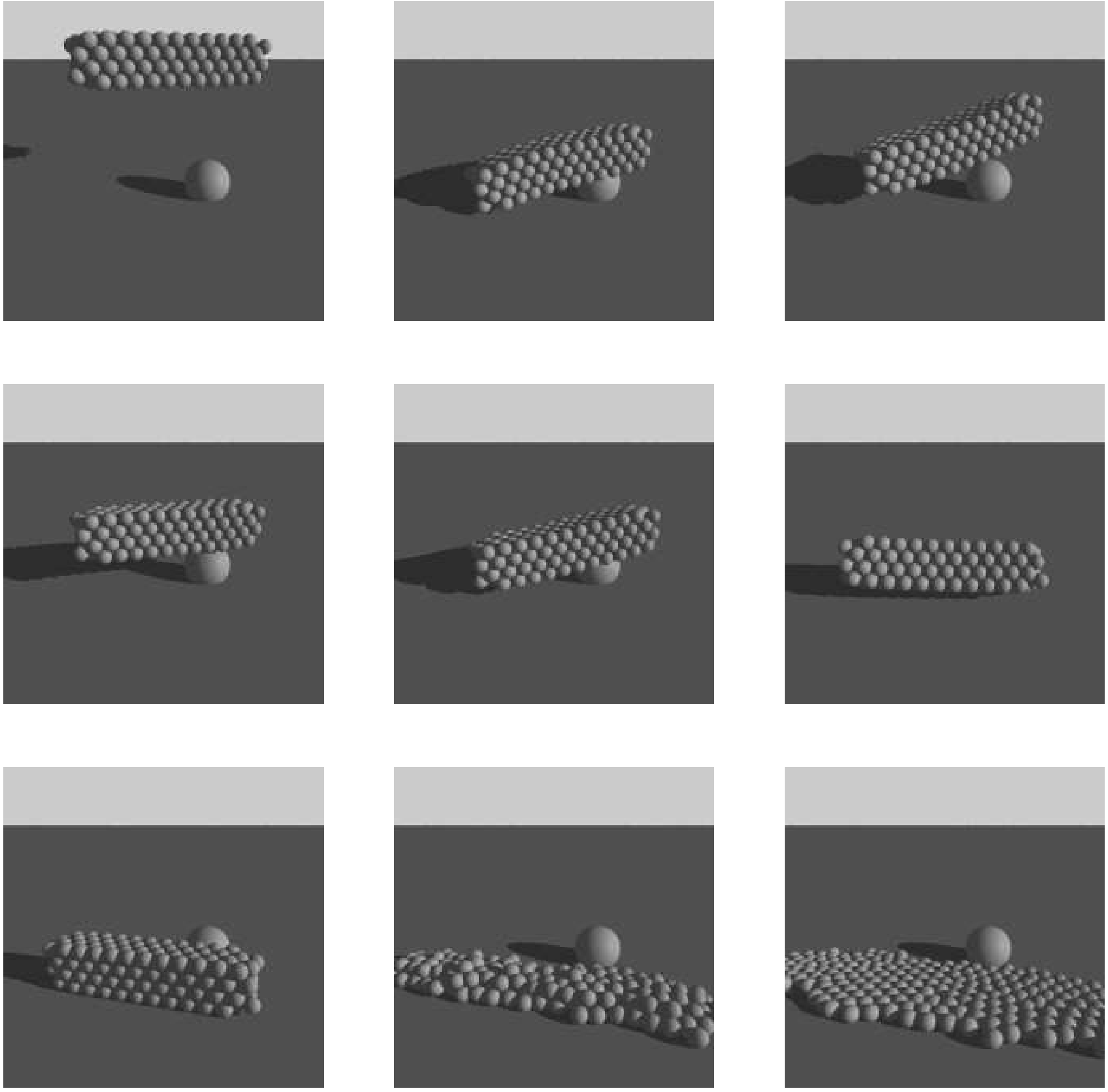


Figure 8.3: Beam colliding and melting

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.

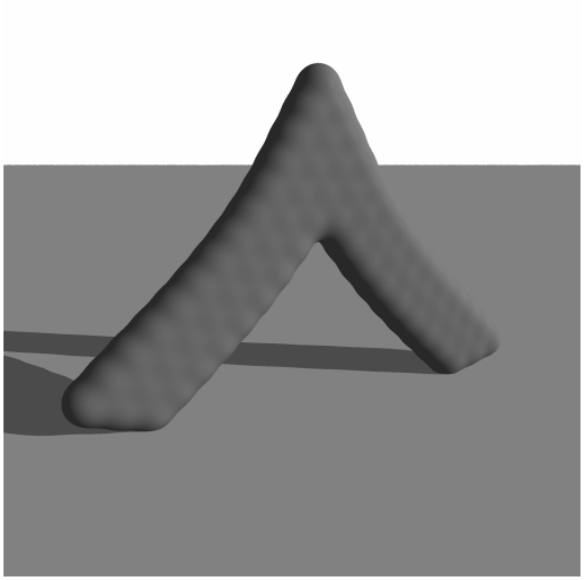


Figure 8.4: Flexible solid

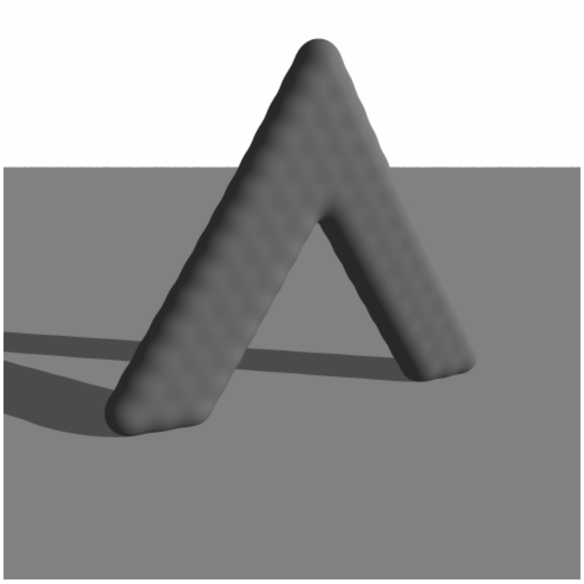


Figure 8.5: Rigid solid

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 $(n = 8, m = 6)$.

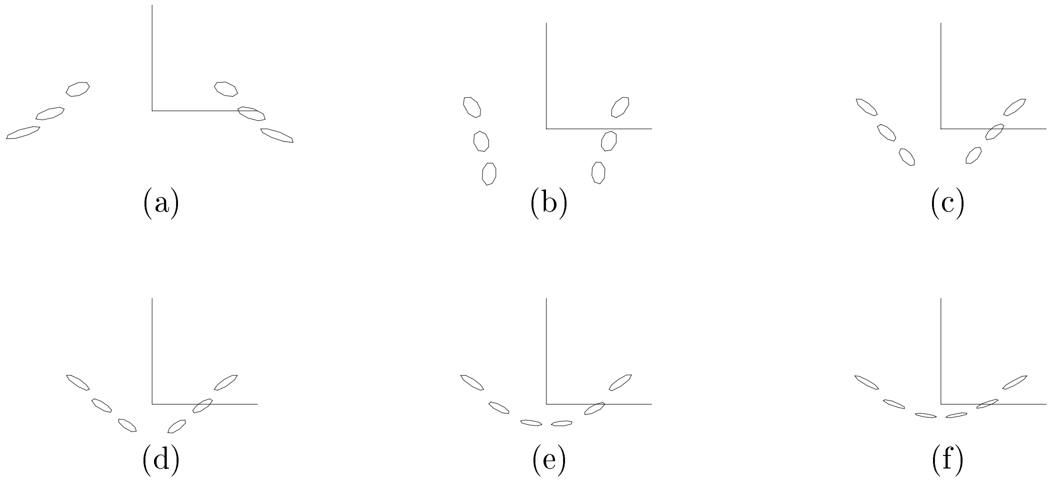


Figure 8.6: Self joining chain under gravity

"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).

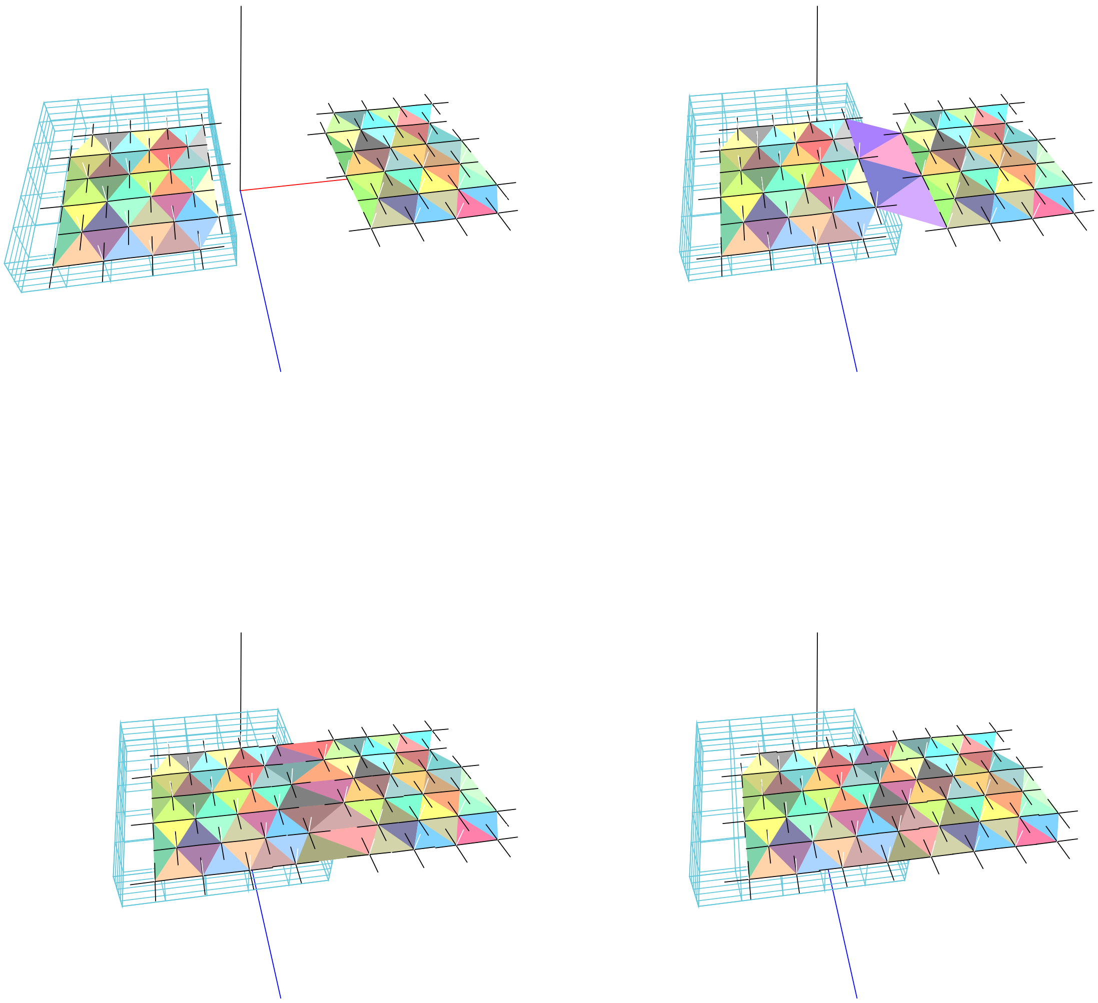


Figure 8.7: Welding two surfaces together.

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

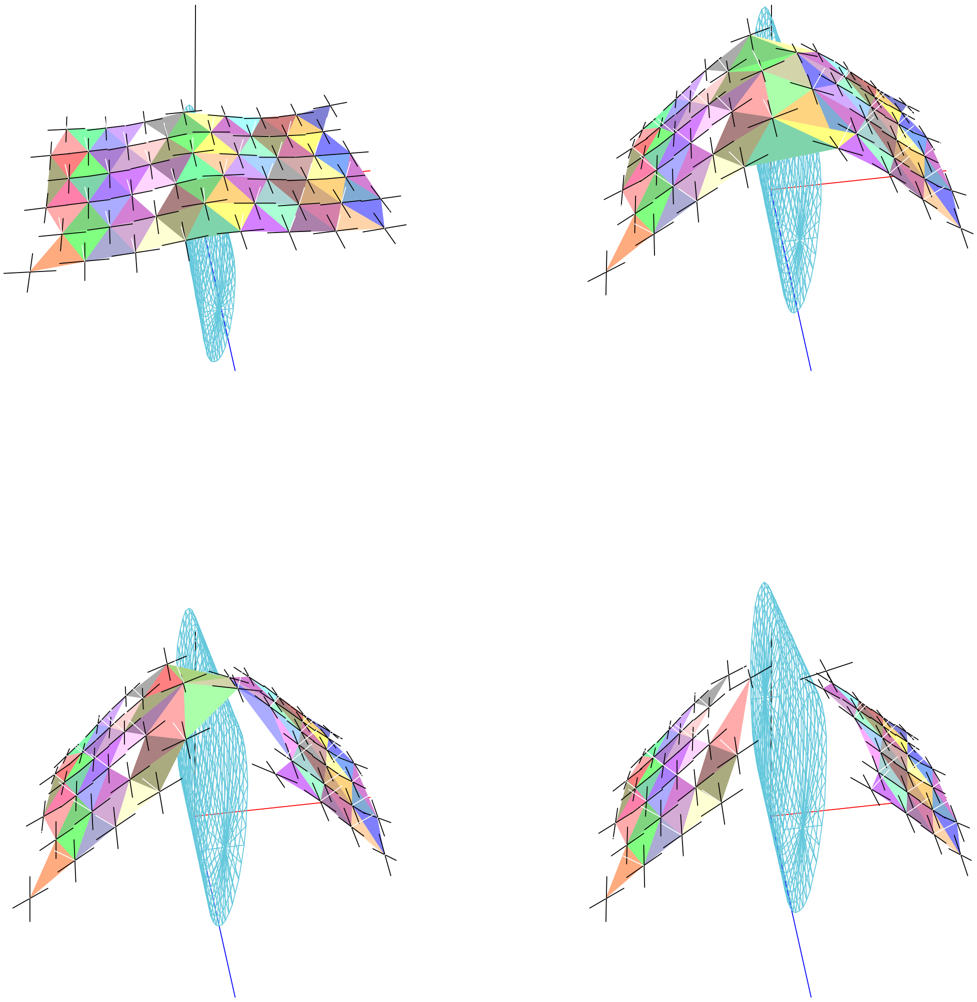


Figure 8.8: Cutting a surface into two.

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.

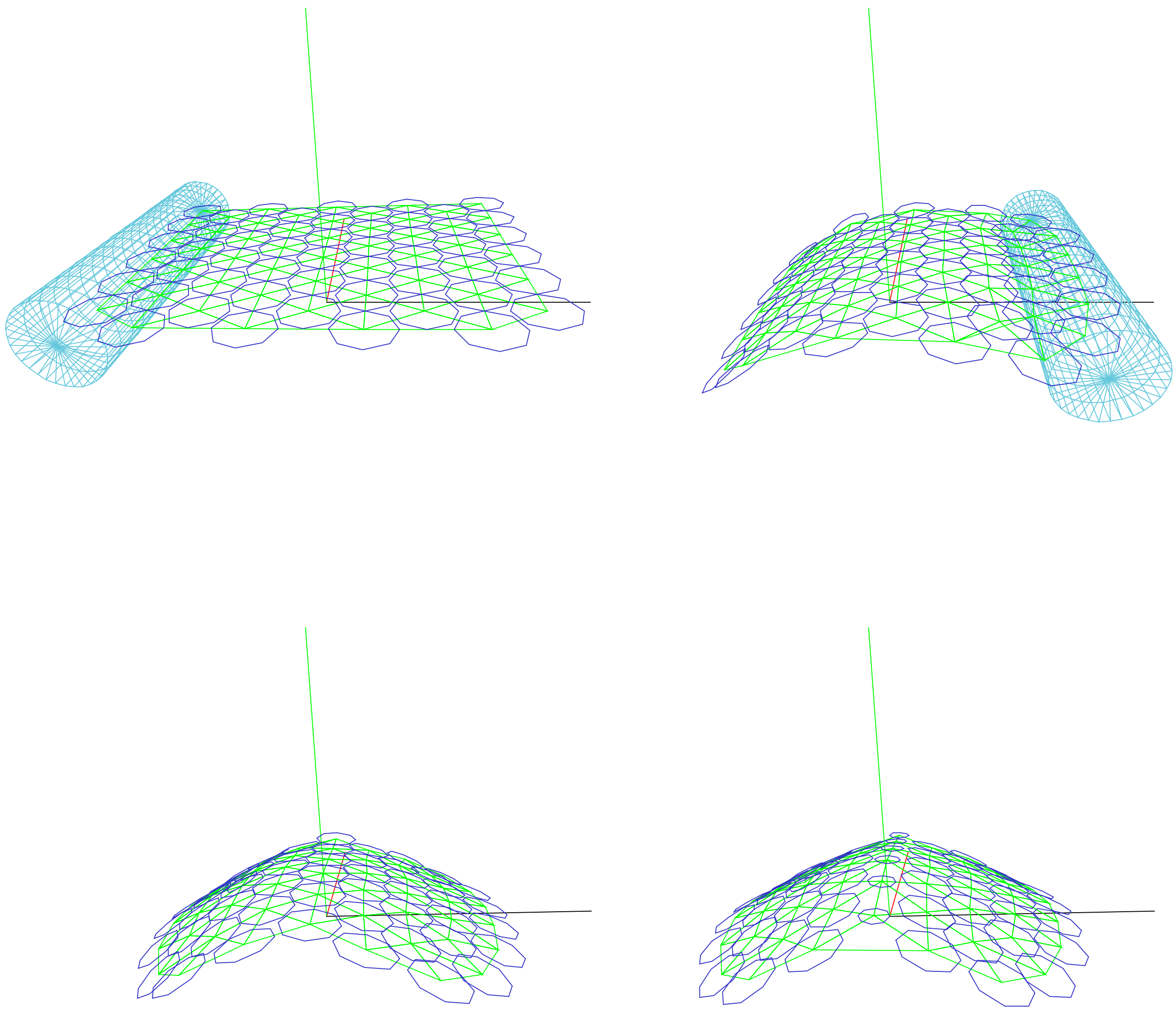


Figure 8.9: Putting a crease into a surface.

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

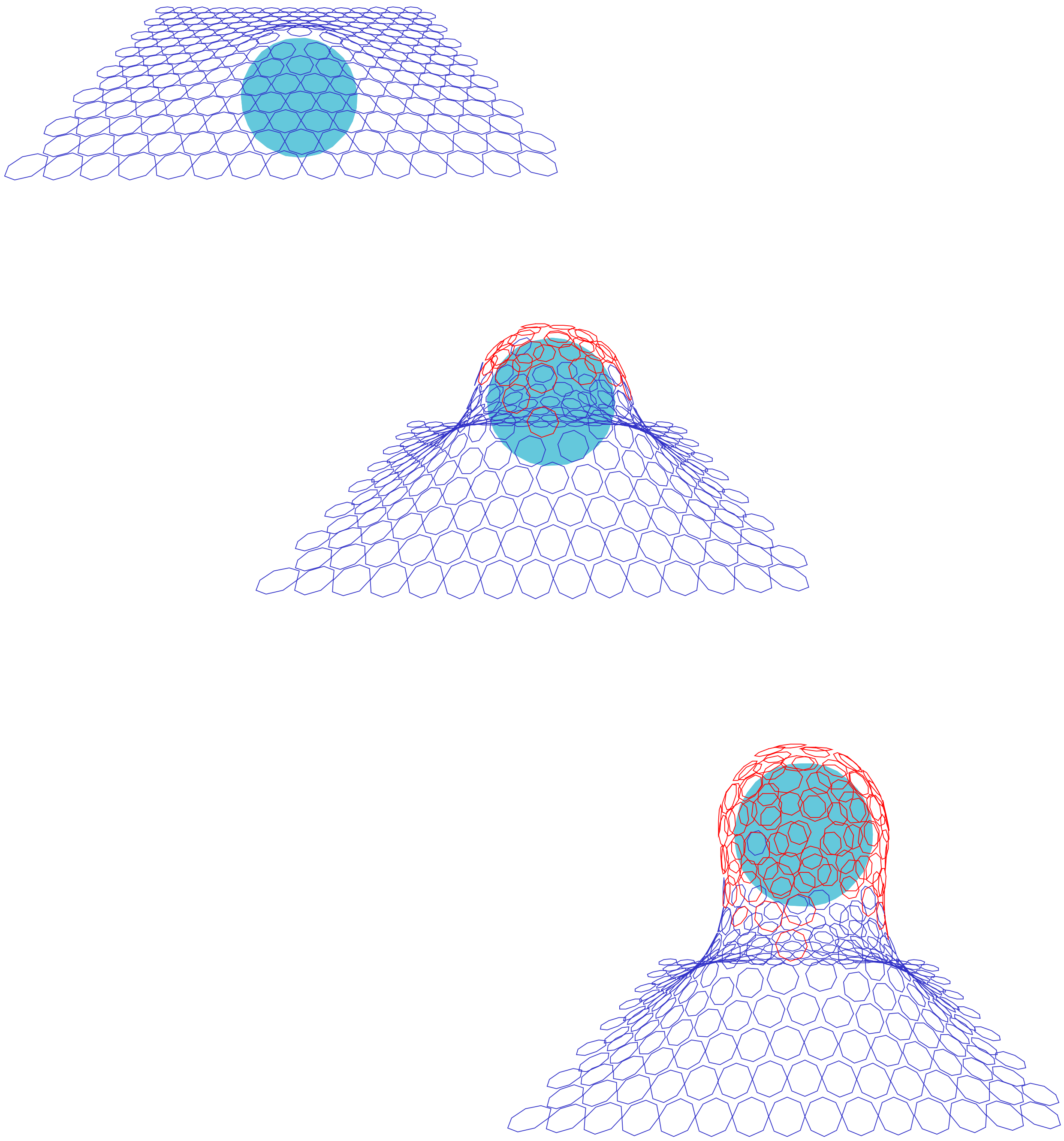


Figure 8.10: Particle creation during stretching

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

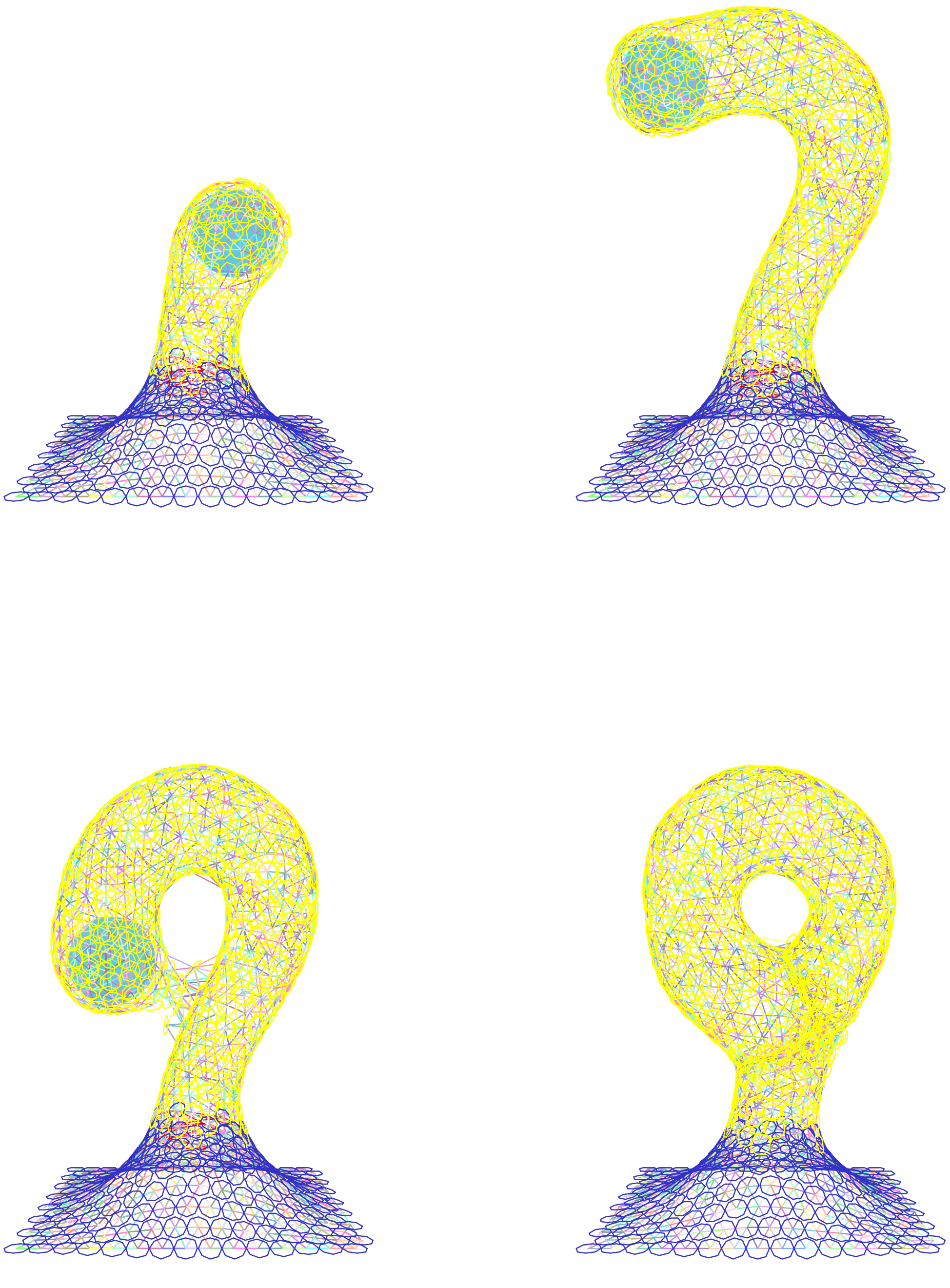


Figure 8.11: Forming a complex object

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

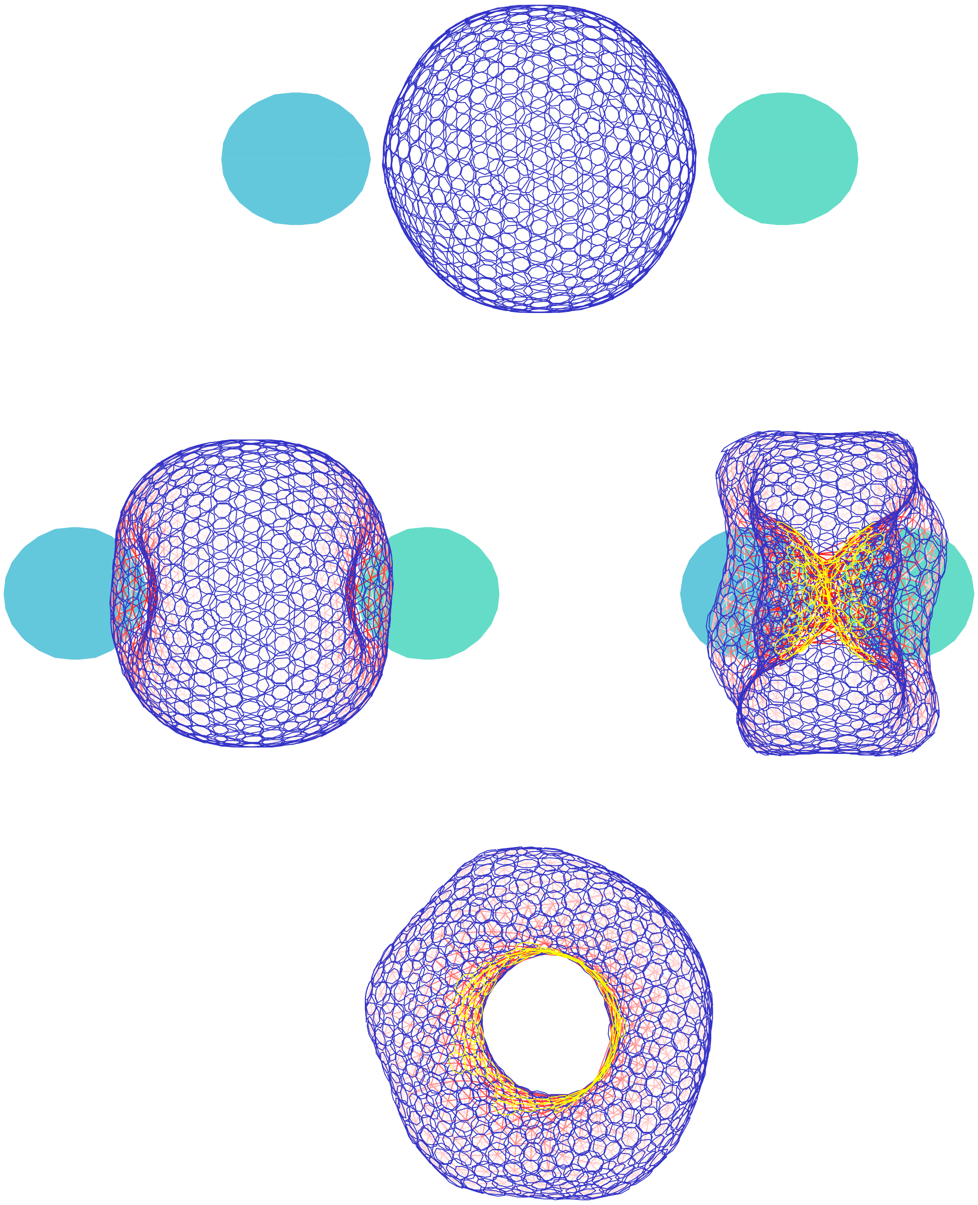


Figure 8.12: Deformation of sphere to torus

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

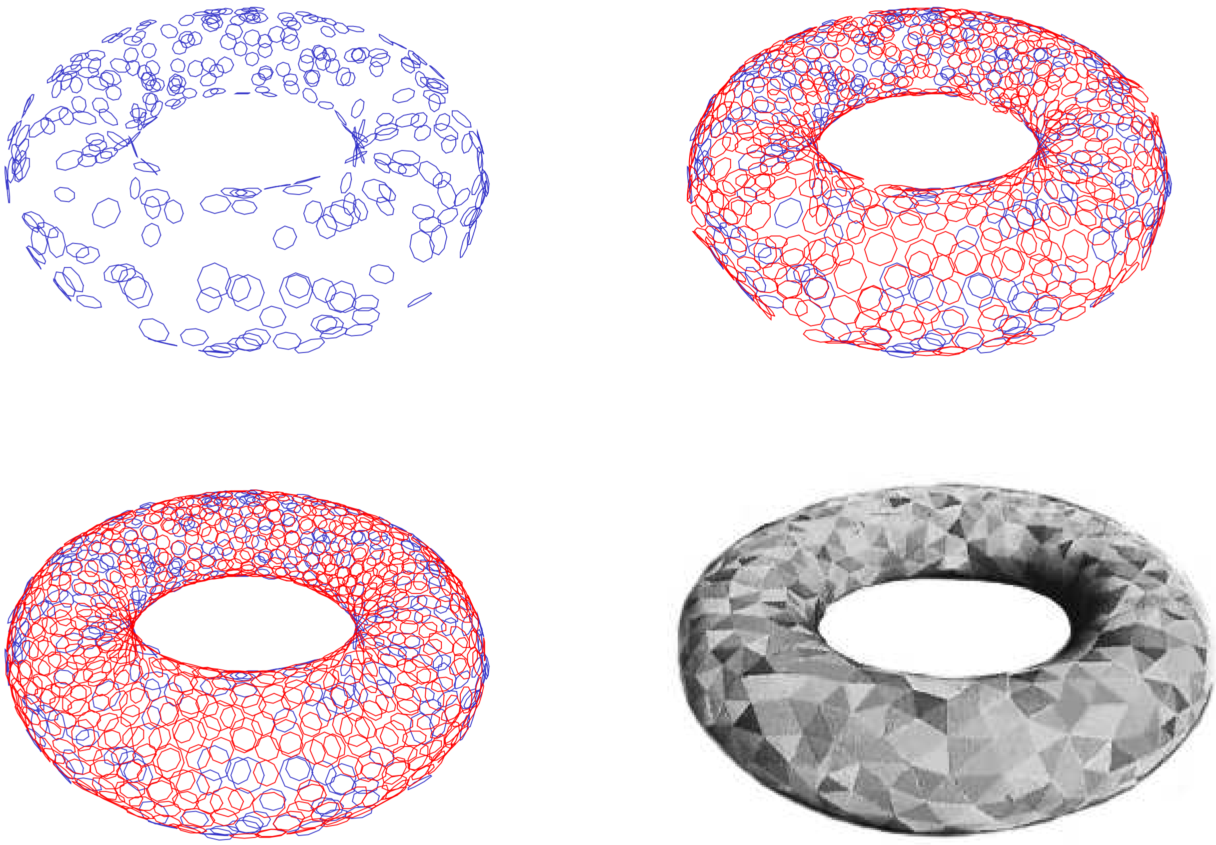


Figure 8.13: Surface interpolation through 3D points

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.

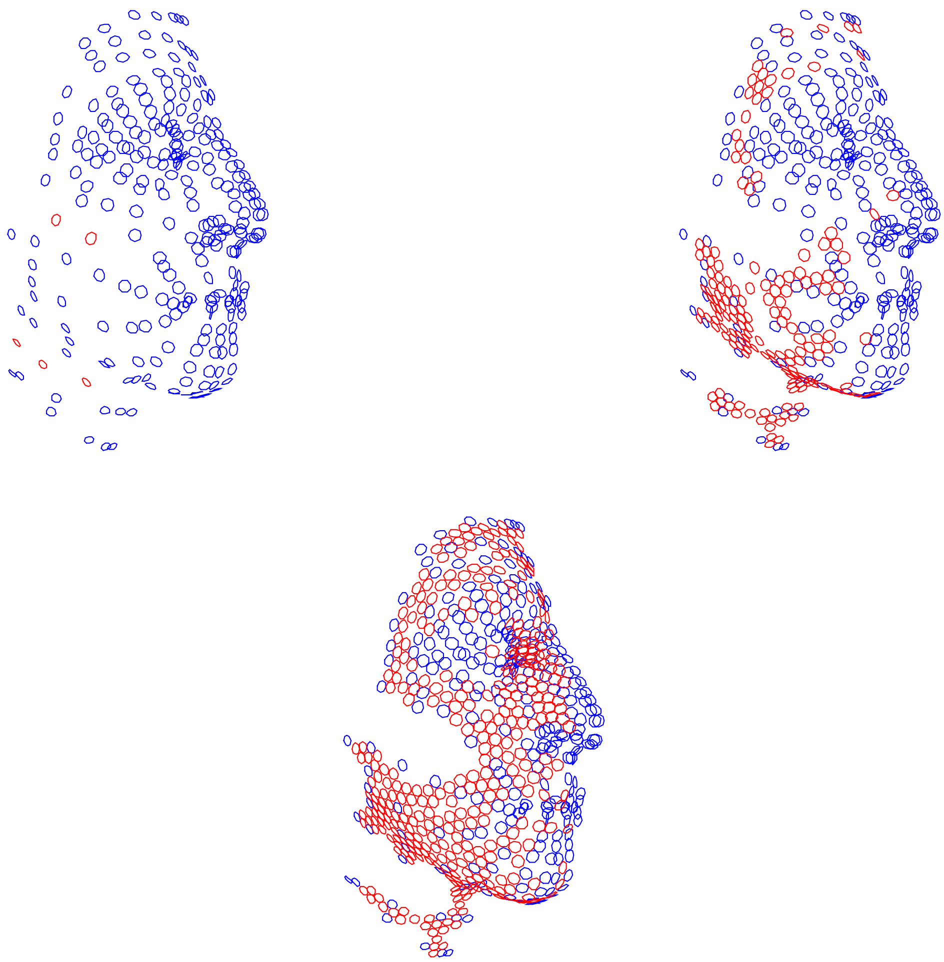


Figure 8.14: Interpolation through 3D points

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.

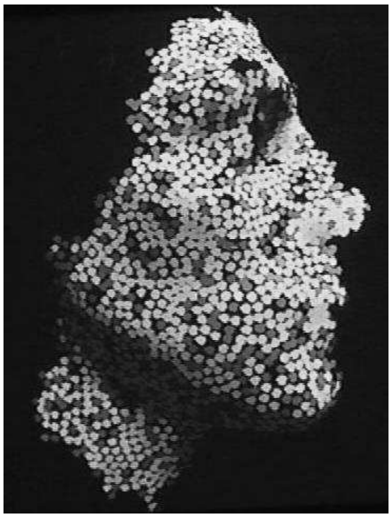


Figure 8.15: Interpolation with increased range

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

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

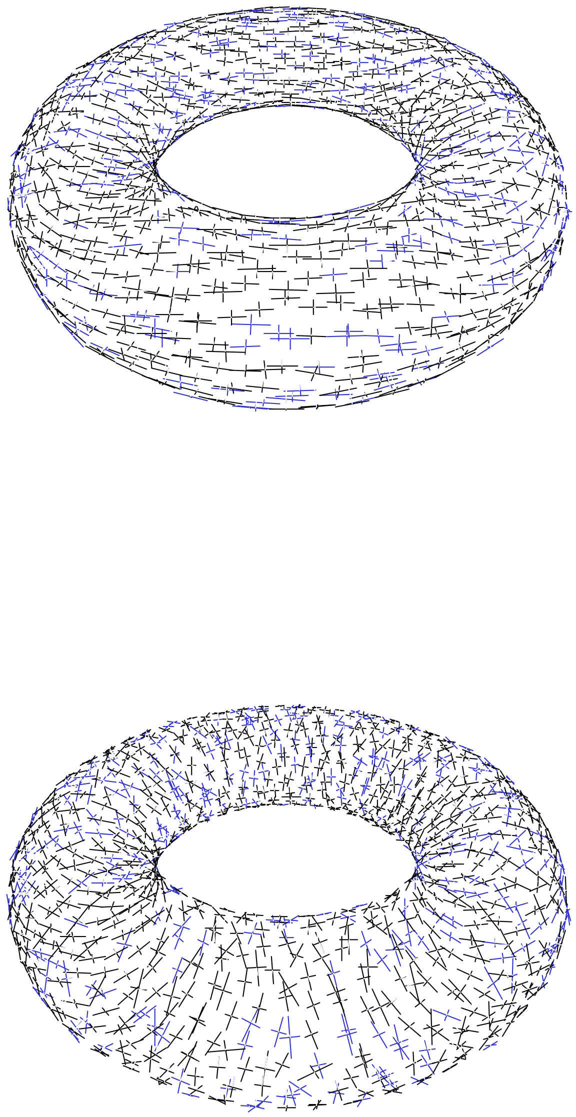


Figure 8.16: Lines of curvature

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).

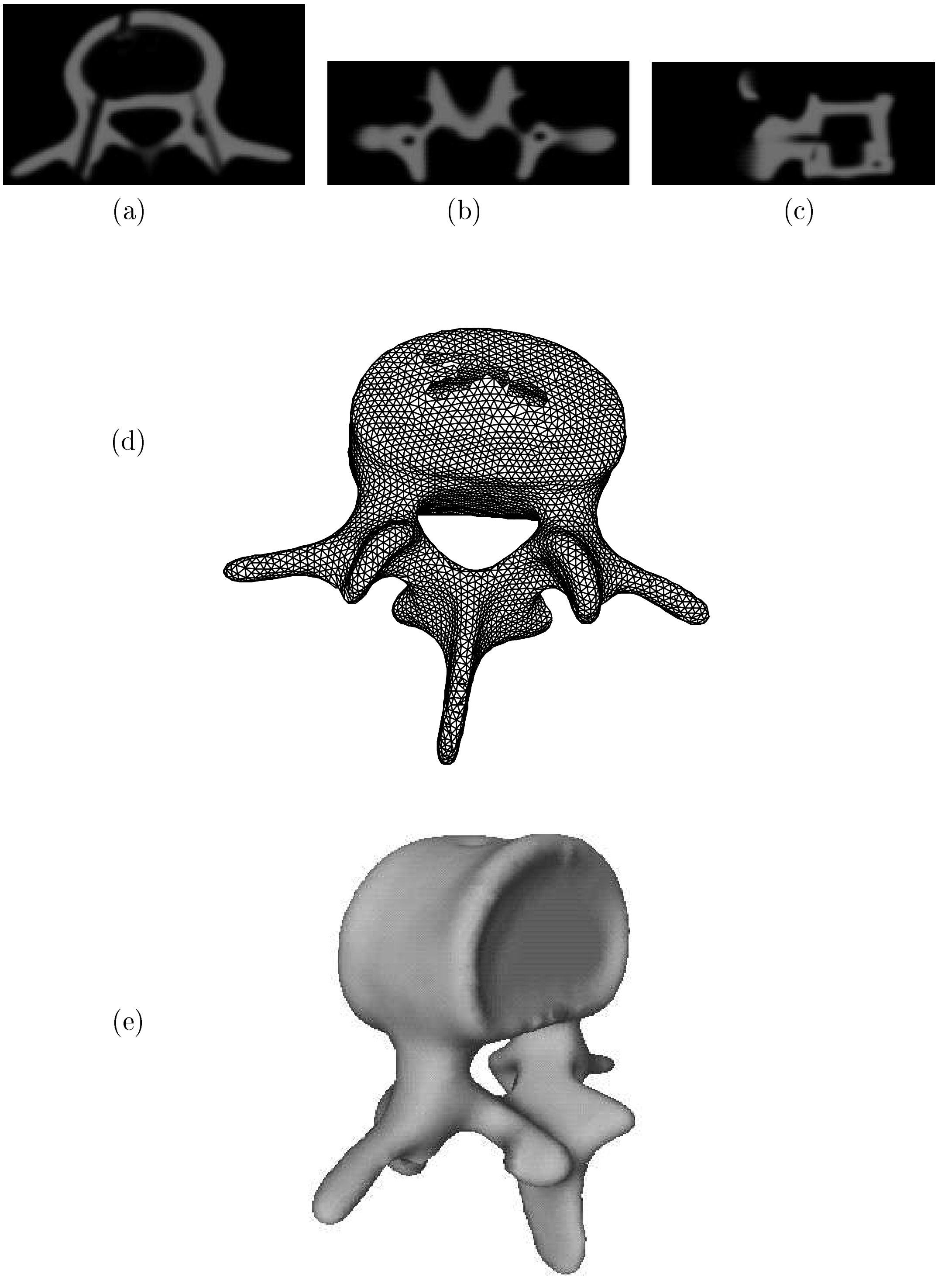


Figure 8.17: Reconstruction from CT volume data

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.

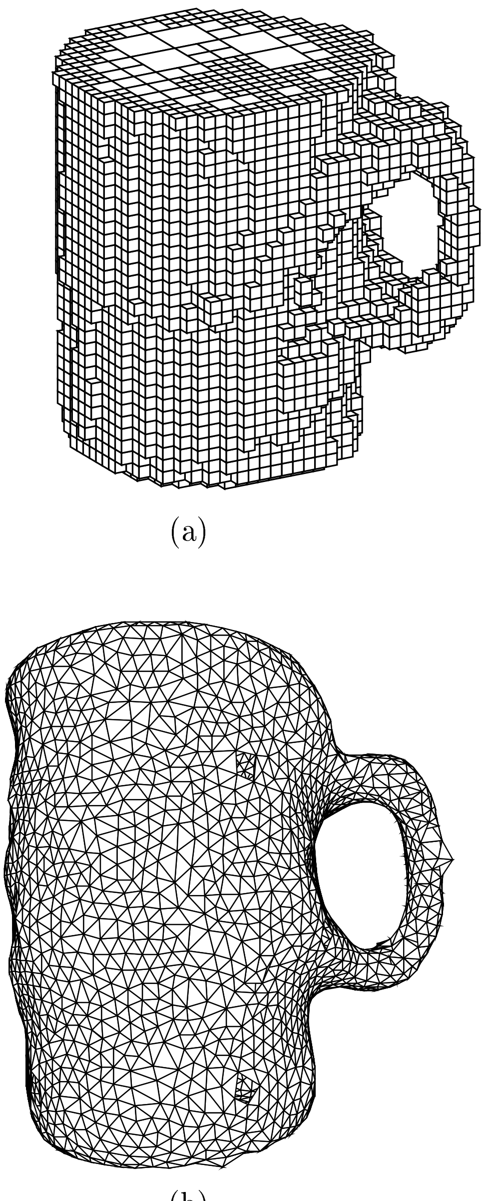
8.4 Summary

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.

Chapter 9

Conclusions

The modeling, description, and animation of shape is a central issue in computer graphics. Our new model, based on dynamically coupled particle systems that interact according to physically inspired potential functions, is free of topological restrictions. Our model has characteristics of traditional spline models, physically-based surface models, and of particle systems. It can be used to model smooth, elastic, moldable surfaces, like traditional splines. Like physically based surface models, by adjusting the relative strengths of various potential functions, a surface's resistance to stretching, bending, or variation in curvature can all be controlled. And yet it allows for arbitrary topologies, like particle systems.

The ability to split, join, and cut are critical to the free-form modeling of shape. Particle based surfaces have the distinct advantage of being easy to shape, extend, join, and separate. For example, the surfaces can be joined and cut at arbitrary locations. Previous approaches, such as spline based models and deformable models, require manual discretization of the surface into patches (for spline based surfaces) or a specification of connectivity (for spring-mass systems) to accomplish these operations. In addition, the ability to easily change topology greatly simplifies the creation and animation of topologically complex surfaces. Unlike previous surface models, the local connections in our system are determined automatically rather than through manual intervention. It is through the interaction of a collection of primitive elements that global shape emerges.

Our geometric shaping tools are analogous to tools used to shape physical objects. These tools allow for intuitive interaction in which topologically complex surfaces can be easily created and modified. In addition, the ability to grow new particles gives our models more fluid-like properties which extend the range of interactions. Like previous deformable surface models, our particle-based surfaces can simulate flexible materials such as cloth, elastic sheets, and plastic films. The ability of our models to automatically connect and disconnect in response to forces provides for the physically inspired animation of models in which the topology changes over time. The combination of these characteristics make particle-based surfaces a powerful new tool for the interactive construction, modeling, and animation of free-form surfaces.

The flexibility of particle based modeling comes at a cost. A limitation of particle-based surfaces is that it is harder to achieve exact analytic (mathematical) control over the shape of the surface. For example, the reconstructed torus is not circularly symmetric, due to the discretization effects of the relatively small number of particles. This behavior could be remedied by adding 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; thus the latter may 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.

Our demonstrations to date have been limited to 2D pointing (input) devices and a modest number of particles. Further investigation into the use of true 3D input devices, preferably with force feedback, would complement the flexibility and intuitive nature of this modeling technique. Implementing these two ideas on the more powerful computers of the future could result in "computational modeling clay".

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. Particle systems can compute complete, detailed, viewpoint invariant geometric surface descriptions. Both open and closed surfaces can be reconstructed, both with and without holes. We can also use our surfaces to segment 3-D volumetric data, and to incrementally construct 3-D object models from motion sequences. Unlike most other deformable models, the topology of the surface need not be initially specified. Because of the flexibility of the technique, and because it is an optimal surface fitting approach, we believe this approach will form the basis of a powerful new class of shape models for computer vision applications.

We have demonstrated that we can interactively shape, extend, join, and separate, particle based surfaces (Figure 8.7). Using shaping tools, we can create topologically complex surfaces (Figure 8.11). We have shown that particle based surfaces simulate a wide range of behaviors, such as the draping of cloth, the stretching of elastic surfaces, the flexibility of thin plastic films, the ability to break or tear, fluid like behavior, the variation of physical properties based a temperature, the diffusion of heat, and interaction with other objects. We have interpolated both open and closed surfaces through collections of sparse 3D points and segmented scalar 3D data. The surfaces are viewpoint invariant and support optimal fitting based on the selected energy functions. A reconstructed surface model can be used as the starting point to interactively create a new shape, manipulate, and then animate it within a virtual environment. Thus particle systems provide a powerful new interface between surface reconstruction in computer vision, free form modeling in computer graphics, and computer assisted animation.

Appendix A

Differential Geometry

Differential geometry is the mathematical study of intrinsic shape (Kreyszig, 1959; Lord and Wilson, 1984; Koenderink, 1990; Farin, 1992; Gray, 1993). In this appendix we introduce the differential geometry of three dimensional space curves and surfaces embedded in three dimensions.

A.1 The Geometry of Curves

A three dimensional space curve can be thought of as the locus of a point moving through space. The movement can be expressed as a function of a single parameter, such as $\mathbf{x}(t) = (x(t), y(t), z(t))$ where t is a real number, and $x(t)$, $y(t)$, and $z(t)$ are single valued functions. Instead of focusing on the parametrization of the curve we will discuss the shape of the curve, that is the geometric properties of the curve, in terms of the distance traveled along the curve.

A.1.1 Arc Length

The distance traveled in moving along a curve, from say $t = a$ to $t =$b, is given by (Kreyszig, 1959)

$$s = \int_a^b \left\| \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \right\| dt = \int_a^b \left(\frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}t} \right)^{\frac{1}{2}}$$

which can be written symbolically as

$$ds^2 = d\mathbf{x} \cdot d\mathbf{x}. \tag{A.1}$$

ds is called the *arc element* and s is the *arc length.* Note that arc length is independent of the choice of parametric representation. Thus geometric measures based on the arc length are invariant to parametrization.

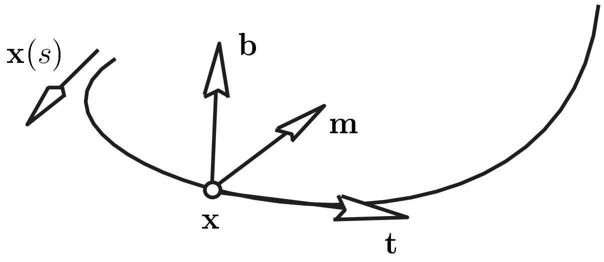


Figure A.1: The Frenet frame.

A.1.2 The Frenet Frame

The first derivative of a curve with respect to arc length defines the unit vector that is tangent to the curve at the point under consideration

$$\dot{\mathbf{x}} = \mathbf{t},$$

where over-struck dot denotes a derivative with respect to s. We call this the *unit tangent vector*. The second derivative with respect to arc length

$$\ddot{\mathbf{x}} = \kappa\mathbf{m}$$

yields a vector with magnitude κ. The unit vector $\mathbf{m}$ is called the *principal normal* and κ the curvature. The tangent and principal normal vectors are orthogonal. Taking the vector product of the tangent and the principal normal yields the unit *bi-normal* vector

$$\mathbf{b} = \mathbf{t} \times \mathbf{m}$$

resulting in a frame at point $\mathbf{x}$, as shown in Figure A.1. This frame, called the *Frenet frame*, describes the local properties of the curve. The first derivative of the bi-normal with respect to arc length

$$\dot{\mathbf{b}} = -\tau\mathbf{m}$$

yields a vector in the direction of the principal normal with magnitude of τ. The scalar τ is the torsion. The curvature, and torsion describe the rotation of the Frenet frame as it moves along the curve in direction s.

A.1.3 The Osculating Circle

The plane spanning the tangent and principal normal vectors is the *osculating plane*. In this plane, there exists a unique circle that is tangent to the curve and with second order continuity matching the curve at $\mathbf{x}(s)$. The circle is named the *osculating circle* and its radius the *radius of curvature*. The fact that it is second order continuous means that the rate of change of the circle's and the curve's tangent vectors, match. This measure of change is the curvature κ and is equal to the inverse of the radius ρ of the circle, that is $\rho = 1/\kappa$. Thus, for straight lines, the curvature is zero and the radius of curvature is infinite. The *osculating circle* is shown in Figure A.2.

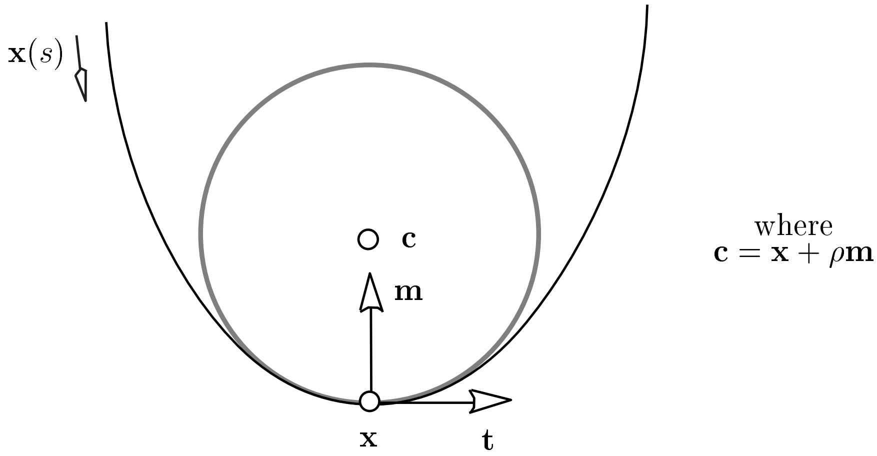


Figure A.2: The osculating circle

As the frame moves along the curve the frame changes position and orientation. One can think of the curvature as the angular velocity (Farin, 1992) of the rotation of the tangent vector, per arc length. The rotation is in the osculating plane, in the direction of the normal vector. The torsion τ is the angular velocity of the bi-normal vector, which twists about the tangent.

A.1.4 The Frenet-Serret formulas

The *Frenet-Serret* formulas (Kreyszig, 1959; Koenderink, 1990; Farin, 1992) describe the changes in the Frenet frame in terms of the frame itself,

$$\begin{bmatrix} \dot{\mathbf{t}} \\ \dot{\mathbf{m}} \\ \dot{\mathbf{b}} \end{bmatrix} = \begin{bmatrix} 0 & \kappa & 0 \\ -k & 0 & \tau \\ 0 & -\tau & 0 \end{bmatrix} \begin{bmatrix} \mathbf{t} \\ \mathbf{m} \\ \mathbf{b} \end{bmatrix}. \tag{A.2}$$

A.2 The Geometry of Surfaces in 3D

A surface may be described by a regular parametrization of position

$$\mathbf{x} = \mathbf{x}(u, v) = \begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix},$$

where the coordinates x, y, and z are differentiable functions of the two real variables u and v. To allow a succinct mathematical description of the geometrical properties of the surface, we adopt the notation where subscripts denote partial derivatives with respect to the parameters u and v. For the first partial derivatives we use the notation

$$\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u} \qquad \mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}.$$

For the second partial derivatives we use the notation

$$\mathbf{x}_{uu} = \frac{\partial^2 \mathbf{x}}{\partial u^2} \qquad \mathbf{x}_{vv} = \frac{\partial^2 \mathbf{x}}{\partial v^2} \qquad \mathbf{x}_{uv} = \frac{\partial \mathbf{x}}{\partial u}\frac{\partial \mathbf{x}}{\partial v} = \frac{\partial^2 \mathbf{x}}{\partial u \partial v} = \frac{\partial^2 \mathbf{x}}{\partial v \partial u} = \mathbf{x}_{vu}.$$

A.2.1 The Arc Element

Given two points on the surface $\mathbf{x}(u, v)$. The vector $d\mathbf{x}$ connecting the two points is given by

$$d\mathbf{x} = \mathbf{x}_u du + \mathbf{x}_v dv.$$

The distance on the surface (Lord and Wilson, 1984) between two such points is

$$ds^2 = d\mathbf{x} \cdot d\mathbf{x} = Edu^2 + 2Fdudv + Gdv^2, \tag{A.3}$$

where

$$E = (\mathbf{x}_u)^2 \qquad F = \mathbf{x}_u \cdot \mathbf{x}_v \qquad G = (\mathbf{x}_v)^2.$$

This is a direct result of equation (A.1). The *arc element*, ds, is a geometric invariant of the surface and thus does not depend on the chosen parametrization, similar to the 3D curve analysis. Equation A.3 for the squared arc element is called the *first fundamental form of the surface* in classical differential geometry.

A.2.2 Tangents and Normal

The vectors tangent to the surface in the directions of the parametrization u and v are given by the partial derivatives $\mathbf{x}_u$ and $\mathbf{x}_v$ respectively. The vector

$$\mathbf{x}_u \times \mathbf{x}_v$$

is orthogonal to the tangent vectors and thus normal to the surface. The magnitude of the normal vector (Kreyszig, 1959) is

$$\|\mathbf{x}_u \times \mathbf{x}_v\| = (\mathbf{x}_u)^2(\mathbf{x}_v)^2 - (\mathbf{x}_u \cdot \mathbf{x}_v)^2 = EG - F^2,$$

so the unit normal is given by

$$\mathbf{n} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{(EG - F^2)^{\frac{1}{2}}}.$$

Figure A.3 shows a local portion of a parametric surface. The dashed lines indicate curves on the surface of constant u and v value. The tangential vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ are the rate of change of the surface along the lines of constant v and u respectively, and the vector $\mathbf{n}$ is normal to the surface.

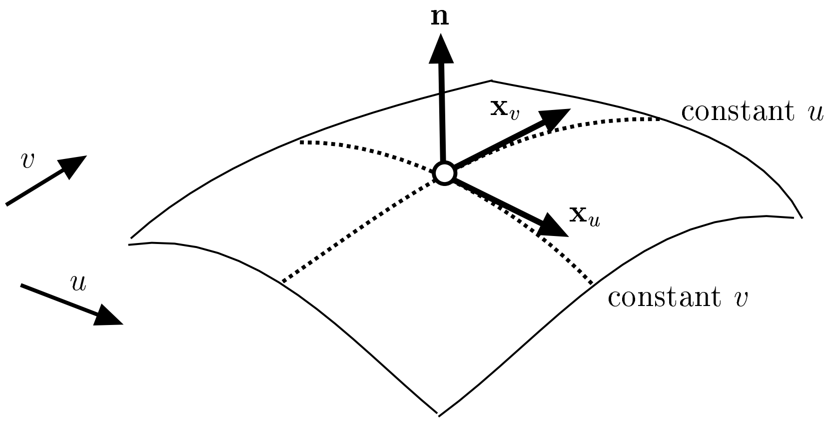


Figure A.3: Surface $\mathbf{x}(u, v)$ with partial derivatives and normal.

A.2.3 The Second Fundamental Form

The first fundamental form (A.3) determines the intrinsic geometrical properties of the surface and is independent of the embedding space. To specify the embedding in Euclidean 3-space requires additional information. That information is the way the *normal* to the surface varies from surface point to surface point. The difference in unit normal at two infinitesimally close points is

$$d\mathbf{n} = \mathbf{n}_u du + \mathbf{n}_v dv.$$

The *second fundamental form* (Kreyszig, 1959; Farin, 1992) is

$$-d\mathbf{n} \cdot d\mathbf{x} = Ldu^2 + 2Mdudv + Ndv^2, \tag{A.4}$$

where

$$L = -\mathbf{n}_u \cdot \mathbf{x}_u = +\mathbf{n} \cdot \mathbf{x}_{uu} = \mathbf{n} \cdot \mathbf{x}_{uu} \tag{A.5}$$
$$M = -\mathbf{n}_u \cdot \mathbf{x}_v = -\mathbf{n}_v \cdot \mathbf{x}_u = \mathbf{n} \cdot \mathbf{x}_{uv} \tag{A.6}$$
$$N = -\mathbf{n}_v \cdot \mathbf{x}_v = +\mathbf{n} \cdot \mathbf{x}_{vv} = \mathbf{n} \cdot \mathbf{x}_{vv} \tag{A.7}$$

The equality found in M comes from differentiating the identities $\mathbf{n} \cdot \mathbf{x}_u = 0$ and $\mathbf{n} \cdot \mathbf{x}_v = 0$ with respect to u and v. The first and second fundamental forms together determines the shape of the surface uniquely; however they do not specify global position and orientation.

A.2.4 Normal Curvature

The first and second fundamental forms determine the shape of the surface local to each point on the surface. Consider a 3D space curve on the surface passing through the surface point $\mathbf{x}$. The relation

$$\kappa \cos\theta = \frac{-d\mathbf{n} \cdot d\mathbf{x}}{ds^2} \tag{A.8}$$

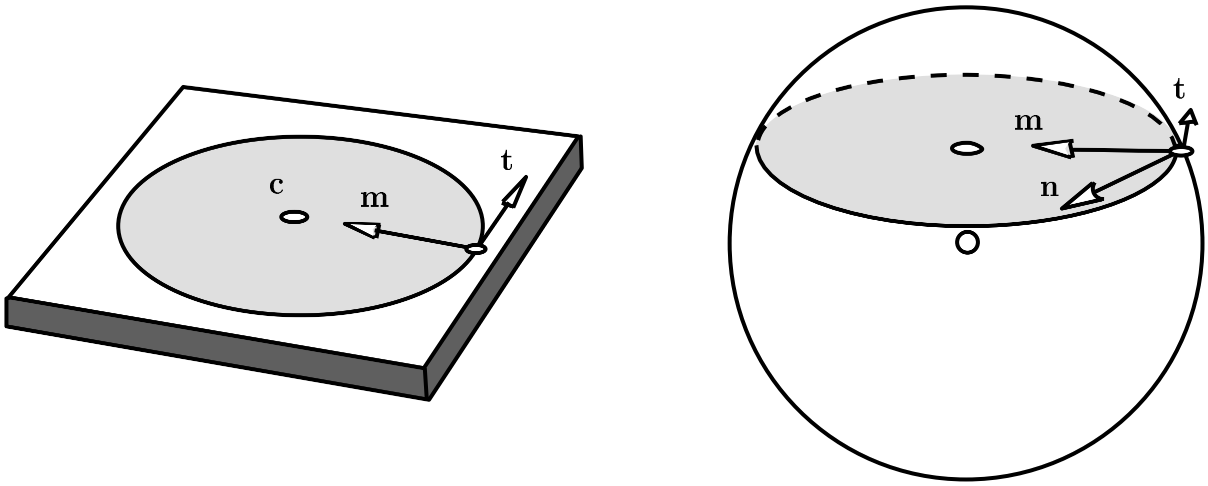


Figure A.4: Curve on sphere where $\mathbf{m} \neq \mathbf{n}$.

relates the curvature κ of the curve to the angle θ between the curve normal $\mathbf{m}$ to the surface normal $\mathbf{n}$. While at first one might expect the normals to match identically one can easily see that this is not necessarily the case, as shown in Figure A.4. Consider a circle as a space curve. The normal $\mathbf{m}$ of the curve is always pointing inward toward the center of the circle. For a curve that is a circle, the osculating circle of the curve is identically the curve. Now consider a sphere where the surface normal $\mathbf{n}$ is always pointing inward to the sphere's center. For circles lying on the sphere, only circles that are great circles of the sphere have normals that match the normals of the sphere. The angle between the curve normal and the surface normal is described by (A.8).

By Meusnier's theorem (Farin, 1992), the osculating circles of all surface curves passing through a point $\mathbf{x}$ and having the same tangent $\mathbf{t}$ form a sphere. This sphere and the surface share a common tangent plane at $\mathbf{x}$, and the sphere has a unique center of curvature, its center. Thus to describe the shape of the surface it is sufficient to study a subset of these curves; namely the curves at $\mathbf{x}$ for which $\mathbf{m} = \mathbf{n}$. There exists one such curve for each tangent vector $\mathbf{t}$. When the osculating plane of a surface curve passing through point $\mathbf{x}$ is perpendicular to the surface tangent plane, then $\theta = 0$ and $\mathbf{m} = \mathbf{n}$. Such curves are called *normal sections* and can be thought of as the intersection of the surface with a plane normal to the surface and which contains the desired tangent vector $\mathbf{t}$, as shown in Figure A.5. The curvature of a normal section at a point $\mathbf{x}$ is the *normal curvature* κ_n and is given by (A.8) with $\theta = 0$

$$\kappa_n = \frac{-d\mathbf{n} \cdot d\mathbf{x}}{ds^2} = \frac{Ldu^2 + 2Mdudv + Ndv^2}{Edu^2 + 2Fdudv + Gdv^2}.$$

A.2.5 More Curvature Measures

In differential geometry curvature is measured in a variety of ways. For completeness we will quickly review some of the more common measures.

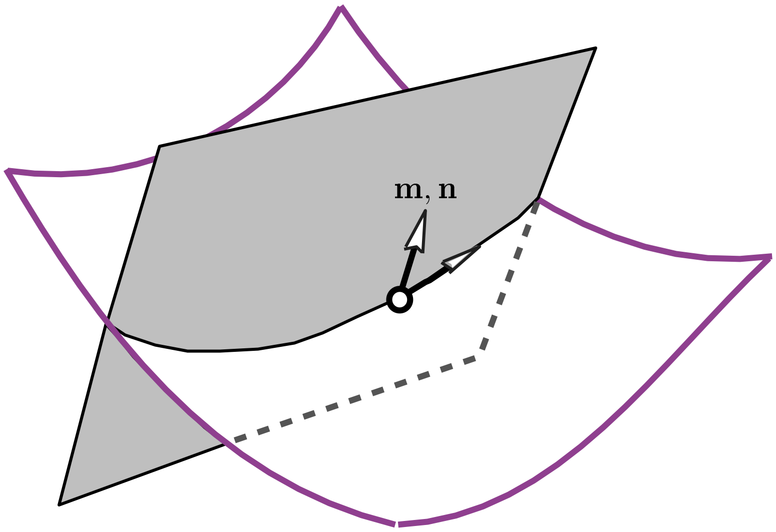


Figure A.5: The normal section.

The extreme values of normal curvature, κ_1 and κ_2, at a given point are called the *principal curvatures*. The unit tangent vectors, $\mathbf{e}_1$ and $\mathbf{e}_2$, for which the extreme values occur are called the *principal vectors*. The corresponding directions are called the *principal directions*. When the principal curvatures are non equal the principal vectors are unique and perpendicular. A *principal frame* (Koenderink, 1990) is defined as a frame with major axes matching the surface normal $\mathbf{n}$ and the principal vectors $\mathbf{e}_1$ and $\mathbf{e}_2$.

The arithmetic mean of the principal curvatures, $H = \frac{1}{2}(\kappa_1+\kappa_2)$, is called the *mean curvature*. The *Gaussian curvature* is the product of the extreme curvatures, $K = \kappa_1\kappa_2$. The measures K and $|H|$ are invariant with respect to coordinate transform (Kreyszig, 1959).

Appendix B

Newtonian Dynamics

This appendix reviews the basic theory of Newtonian dynamics and the mathematics necessary for both un-oriented and oriented particle systems. For completeness we occasionally repeat material presented earlier in the dissertation.

A standard particle is described by its position $\mathbf{x}$ and mass m. An oriented particle is described as a standard particle enhanced with an orientation $\mathbf{R}$ and an inertia tensor $\mathbf{I}$.

B.1 Rotation

Each oriented particle defines both a normal vector $\mathbf{n}_i = z$ and a local tangent plane defined by the local x and y vectors. More formally, we write the state of each particle as $(\mathbf{x}_i, \mathbf{R}_i)$, where $\mathbf{x}_i$ 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 $\mathbf{n}_i$.

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

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

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 (Shoemake, 1985).

B.2 Inertia Tensor

The inertia tensor $\mathbf{I}$ relates the angular momentum vector to the angular velocity vector by a linear transformation. In general $\mathbf{I}$ is represented as a 3x3 matrix,

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}.$$

The inertia tensor is symmetric; that is, $I_{ij} = I_{ji}$. The diagonal elements, I_{xx}, I_{yy}, and I_{zz}, are called the moments of inertia of the object. The off-diagonal elements are known as the products of inertia.

The inertia tensor is defined with respect to an inertial frame; a set of coordinate axes and an origin about which the object rotates. For any choice of origin, for any body, there always exists a set of coordinate axes which diagonalizes the inertia tensor (Marion, 1970). Thus we can write the inertia tensor as

$$\mathbf{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix},$$

where the associated moments I_{xx}, I_{yy} and I_{zz} are the principal moments of inertia. These axes are the *principal axes of inertia.*

For oriented particles we are interested in the property of angular momentum, but are not interested in mimicking the inertia of a particular rigid body. Thus we choose the simple inertia tensor of the form

$$\mathbf{I} = c \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{B.1}$$

where c is some scalar constant. We define our inertia tensor about the particle's local origin with respect to the world coordinate axes. Our choice of inertia tensor is equivalent to a spherical object with its centroid at the particle origin. This choice results in a tensor that is constant over change in particle position and orientation. Using principal axes, as we do, allows us to represent the inertia tensor by a triplet (like a vector) encoding the principal moments of inertia. Our choice simplifies the equations of motion as well as the computation of angular momentum and rotational kinetic energy.

B.3 Velocity

Since we are interested in animating a particle system, we must consider how the position and orientation of the particles change over time. Thus, we write the position and orientation as functions of time: $\mathbf{x}(t)$ and $\mathbf{R}(t)$.

The linear velocity is defined as the rate of change of the particle position over time. At time t the velocity of the particle is

$$\mathbf{v}(t) = \frac{\mathrm{d}\mathbf{x}(t)}{\mathrm{d}t}$$

The instantaneous angular velocity is defined as the rate of change of the orientation over time

$$\boldsymbol{\omega}(t) = \frac{\mathrm{d}\mathbf{R}(t)}{\mathrm{d}t}.$$

The vector properties of addition and scalar multiplication hold for instantaneous angular velocities (Hoffmann, 1966). For example, given two instantaneous angular velocities $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$, the following holds

$$\boldsymbol{\omega}_1 + \boldsymbol{\omega}_2 = \boldsymbol{\omega}_2 + \boldsymbol{\omega}_1.$$

Thus, we can manipulate instantaneous angular velocities as vectors.

We could have derived angular velocity by differentiating the rotation matrix or quaternion but it is simpler to describe it in terms of the rotation $\mathbf{R}(t)$ as shown above. For matrix and quaternion derivations see Baraff (1991). Since we use angular velocity to approximate a rotation over a given time interval, the derivation shown above is ideal for our purposes.

After computing a change in rotation as a vector quantity we then convert it to a quaternion. To update a particle's orientation as a quaternion, we simply form a new unit quaternion $\hat{\boldsymbol{\omega}}$ from the current angular velocity $\boldsymbol{\omega}$ and the time step Δt, and use quaternion multiplication. Details on quaternions can be found in (Shoemake, 1989; Shoemake, 1991).

B.4 Momentum

The linear momentum $\mathbf{p}$ of a particle with mass m and velocity $\mathbf{v}$ is

$$\mathbf{p}(t) = m\mathbf{v}(t).$$

The angular momentum of a particle with inertia $\mathbf{I}$ and angular velocity $\boldsymbol{\omega}$ is

$$\mathbf{L} = \mathbf{I}\boldsymbol{\omega}.$$

The inertia tensor $\mathbf{I}$ relates the angular momentum vector to the angular velocity vector by a linear transformation. The angular momentum simplifies to

$$\mathbf{L} = \begin{bmatrix} I_x\omega_x \\ I_y\omega_y \\ I_z\omega_z \end{bmatrix}$$

when the inertial coordinate frame is aligned with the principal moments. Since we are using the inertia tensor for a spherical object (B.1) the angular momentum simplifies to

$$\mathbf{L} = c\boldsymbol{\omega}$$

for all time values.

B.5 Force and Torque

The force on a particle is the change in momentum which is commonly written in terms of mass and acceleration:

$$\mathbf{f}(t) = \frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \frac{\mathrm{d}m\mathbf{v}(t)}{\mathrm{d}t} = m\frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t} = m\mathbf{a}(t).$$

The torque $\boldsymbol{\tau}$ on a particle is the rate of change in angular momentum and can be written in terms of inertia and angular acceleration:

$$\boldsymbol{\tau}(t) = \frac{\mathrm{d}\mathbf{L}(t)}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{I}\boldsymbol{\omega}(t)}{\mathrm{d}t} = \frac{\mathrm{d}\mathbf{I}}{\mathrm{d}t}\boldsymbol{\omega}(t) + \mathbf{I}\frac{\mathrm{d}\boldsymbol{\omega}(t)}{\mathrm{d}t} = \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} + \mathbf{I}\mathbf{b}(t).$$

For our system, the inertia tensor (B.1) is constant over time and thus the torque reduces to

$$\boldsymbol{\tau}(t) = \mathbf{I}\mathbf{b}(t)$$

for all values of time t.

B.6 Potential Energy

We can derive both forces and torques from potential energy functions. This has the advantage of allowing the design of energy functions which exhibit minimal energies at desired particle configurations, and guarantees that the system will not diverge. A force results from the gradient of the potential energy ϕ with respect to a particle's position $\mathbf{x}_i$,

$$f_i = -\nabla_{\mathbf{x}_i}\phi,$$

and a torque results from the gradient of the potential energy with respect to the particle's orientation θ_i,

$$\tau_i = -\nabla_{\theta_i}\phi.$$

As the energy of the system minimizes, the particles migrate to minimal energy configurations.

Appendix C

Gradients of Potential

This appendix presents mathematical background and identities necessary to derive the force and torque vectors from weighted scalar potential fields. The translational force acting on a particle results from the loss in potential energy due to a change in particle position. Likewise, an angular force, or torque, results from the loss in potential energy due to a change in particle orientation. In three dimensions these are written as the negative gradient of the potential energy function with respect to the particle position and orientation, as follows:

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i} \phi, \tag{C.1}$$

$$\boldsymbol{\tau}_i = -\nabla_{\boldsymbol{\theta}_i} \phi, \tag{C.2}$$

where $\boldsymbol{\theta}_i$ is the infinitesimal change in orientation of particle i.

The six sections discuss fundamental concepts and derive basic identities which can be used to quickly derive the appropriate forces and torques from scalar potentials. Section one presents the multiplication and chain rules of differential calculus applied to the gradient operator. Section two derives the relationship between an infinitesimal rotation and change of the parameters of the potential functions. Section three presents differential identities for the gradient of the Euclidean norm with respect to particle separation and differences in normal vectors. Section four presents differential identities for the gradient of scalar products with respect to combinations of the particle normal and particle separation vectors. Section fives derives the gradient of the weighting function with respect to particle position and orientation. Section six presents identities for deriving the force and torque due to a scalar potential energy function.

C.1 Gradient of Scalar Functions

The derivation of force and torque are based on the application of the multiplication rule and chain rule of differential calculus. If f and g are scalar functions then by the multiplication rule

$$\nabla(fg) = f\nabla g + g\nabla f. \tag{C.3}$$

When f is a function of a scalar variable a then by the chain rule,

$$\nabla f(a) = \frac{\mathrm{d}f}{\mathrm{d}a}\nabla a. \tag{C.4}$$

C.2 Gradient with Respect to Orientation

In this section we derive the gradient with respect to change in infinitesimal orientation of a scalar function, that is $\nabla_{\boldsymbol{\theta}} f$.

Unlike general rotations, infinitesimal rotations behave as vectors. In particular they follow the parallelogram law (addition law) of vectors. Given an infinitesimal rotation about an axis, we can represent the rotation by a vector $\boldsymbol{\theta}$, where the magnitude of the vector represents the angle of rotation, and the direction of the vector $\boldsymbol{\theta}$ points in the direction of the axis of rotation.

C.2.1 The Change in Normal

To understand the relationship between a change in orientation and the corresponding particle normal consider the following. A particle with fixed position and with a vector $\mathbf{n}$ attached, is rotating about its center. As the particle rotates, the vector $\mathbf{n}$ traces a circle about the axis of rotation. The infinitesimal variation in the vector's components may be written, to a first approximation, as the following vector product (Goldstein, 1950)

$$d\mathbf{n} = \mathbf{n} \times d\boldsymbol{\theta}, \tag{C.5}$$

where $d\boldsymbol{\theta}$ is an infinitesimal rotation.

We now introduce matrix-vector notation which allows one to write a vector product as a matrix times a vector. Given two vectors $\mathbf{a}$ and $\mathbf{b}$

$$\mathbf{a} \times \mathbf{b} = \mathbf{A}^{\star}\mathbf{b}$$

where $\mathbf{A}^{\star}$ is given by

$$\begin{bmatrix} 0 & -\mathbf{a}_3 & \mathbf{a}_2 \\ \mathbf{a}_3 & 0 & -\mathbf{a}_1 \\ -\mathbf{a}_2 & \mathbf{a}_1 & 0 \end{bmatrix}.$$

Note that $\mathbf{b} \times \mathbf{a}$ equals $(\mathbf{A}^{\star})^T\mathbf{b}$, but *does not* equal $\mathbf{b}\mathbf{A}^{\star}$.

Rewriting (C.5) in matrix-vector notation we have

$$d\mathbf{n} = \mathbf{n} \times d\boldsymbol{\theta} = \begin{bmatrix} 0 & -\mathbf{n}_3 & \mathbf{n}_2 \\ \mathbf{n}_3 & 0 & -\mathbf{n}_1 \\ -\mathbf{n}_2 & \mathbf{n}_1 & 0 \end{bmatrix} d\boldsymbol{\theta} = \mathbf{N}^{\star} d\boldsymbol{\theta}.$$

Thus the change in vector $\mathbf{n}$ due to the infinitesimal rotation $d\boldsymbol{\theta}$ is

$$\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\boldsymbol{\theta}} = \mathbf{N}^{\star}. \tag{C.6}$$

C.2.2 Gradient of a Scalar Function

Now let us say we have a scalar function $f(\mathbf{n})$. By the chain rule we can differentiate f with respect to the infinitesimal rotation $d\boldsymbol{\theta}$:

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\theta}} = \left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}}\right)\left(\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\boldsymbol{\theta}}\right). \tag{C.7}$$

In order to replace the second term on the right hand side $\mathbf{N}^\star$, we apply the commutative law of multiplication so that

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\theta}} = \left(\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\boldsymbol{\theta}}\right)\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}}\right). \tag{C.8}$$

We can do this because the multiplication between the various components of the two terms follow scalar product rules. We replace the first term of the product with (C.6)

$$\mathbf{N}^\star\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}}\right).$$

By matrix-vector notation this is equivalent to the cross-product

$$\mathbf{n}\times\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}}\right).$$

When computing the differential of a scalar function with respect to a vector, the differential due to each component of the vector is computed separately. In three dimensions, this can be written as the gradient of the function with respect to the differentiating vector, such as,

$$\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}} = \nabla_{\mathbf{n}}f.$$

Thus we can write the change of a scalar potential function f due to rotation of a particle as

$$\nabla_{\boldsymbol{\theta}}f = \mathbf{n}\times(\nabla_{\mathbf{n}}f), \tag{C.9}$$

where $\mathbf{n}$ is the particle normal. This is a valuable identity that we can use in the derivation of torque due to an inter-particle potential function.

C.2.3 Directional Derivatives

The concise notation in the above derivation has hidden many of the details and may be confusing to follow. We now show how we go from (C.7) to (C.8). The directional derivative (Hay, 1953) of a function $f(\mathbf{n})$ at $\mathbf{x}$ in the direction $\mathbf{b}$ is

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial n_x}\frac{\partial n_x}{\partial s} + \frac{\partial f}{\partial n_y}\frac{\partial n_y}{\partial s} + \frac{\partial f}{\partial n_z}\frac{\partial n_z}{\partial s}$$

where $\partial f/\partial s$ is the rate of change of f in the direction $\mathbf{b}$.

We now differentiate the scalar function f with respect to an infinitesimal rotation $\boldsymbol{\theta} = [\theta_x, \theta_x, \theta_x]^T$.

$$\frac{\mathrm{d}f}{\mathrm{d}\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} f = \begin{bmatrix} \frac{\partial f}{\partial \theta_x} \\ \\ \frac{\partial f}{\partial \theta_y} \\ \\ \frac{\partial f}{\partial \theta_z} \end{bmatrix}.$$

Each element of the vector represents a directional derivative. The derivatives are with respect to a change in orientation about the three major axes. Expanding the elements of the vector we have

$$\begin{bmatrix} \frac{\partial f}{\partial n_x}\frac{\partial n_x}{\partial \theta_x} + \frac{\partial f}{\partial n_y}\frac{\partial n_y}{\partial \theta_x} + \frac{\partial f}{\partial n_z}\frac{\partial n_z}{\partial \theta_x} \\ \\ \frac{\partial f}{\partial n_x}\frac{\partial n_x}{\partial \theta_y} + \frac{\partial f}{\partial n_y}\frac{\partial n_y}{\partial \theta_y} + \frac{\partial f}{\partial n_z}\frac{\partial n_z}{\partial \theta_y} \\ \\ \frac{\partial f}{\partial n_x}\frac{\partial n_x}{\partial \theta_z} + \frac{\partial f}{\partial n_y}\frac{\partial n_y}{\partial \theta_z} + \frac{\partial f}{\partial n_z}\frac{\partial n_z}{\partial \theta_z} \end{bmatrix}.$$

Each of the partial derivatives evaluates to a scalar value, thus the order of the product can be interchanged. For example

$$\frac{\partial f}{\partial n_x}\frac{\partial n_x}{\partial \theta_x} = \frac{\partial n_x}{\partial \theta_x}\frac{\partial f}{\partial n_x}.$$

Thus we can write the above vector as the product of a matrix and vector

$$\begin{bmatrix} \frac{\partial n_x}{\partial \theta_x} & \frac{\partial n_y}{\partial \theta_x} & \frac{\partial n_z}{\partial \theta_x} \\ \\ \frac{\partial n_x}{\partial \theta_y} & \frac{\partial n_y}{\partial \theta_y} & \frac{\partial n_z}{\partial \theta_y} \\ \\ \frac{\partial n_x}{\partial \theta_z} & \frac{\partial n_y}{\partial \theta_z} & \frac{\partial n_z}{\partial \theta_z} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial n_x} \\ \\ \frac{\partial f}{\partial n_y} \\ \\ \frac{\partial f}{\partial n_z} \end{bmatrix}.$$

The matrix is equivalent to $d\mathbf{n}/d\boldsymbol{\theta}$, and the vector is equivalent to $df/d\mathbf{n}$, which is the gradient of f with respect to $\mathbf{n}$. Thus

$$\nabla_{\boldsymbol{\theta}} f(\mathbf{n}) = \left(\frac{\mathrm{d}\mathbf{n}}{\mathrm{d}\boldsymbol{\theta}}\right)\left(\frac{\mathrm{d}f}{\mathrm{d}\mathbf{n}}\right) = \mathbf{N}^{\star}\left(\nabla_{\mathbf{n}} f\right) = \mathbf{n} \times \nabla_{\mathbf{n}} f.$$

C.3 Gradient of the Euclidean Norm

The Euclidean norm of a vector is defined as a measure of the magnitude of the vector, and for any vector $\mathbf{a}$ is defined as

$$\|\mathbf{a}\| = (\mathbf{a}\cdot\mathbf{a})^{\frac{1}{2}} = (\mathbf{a}_x^2 + \mathbf{a}_y^2 + \mathbf{a}_z^2)^{\frac{1}{2}}.$$

A *unit vector* is a vector with Euclidean norm of one. For any vector $\mathbf{a}$, the unit vector $\hat{\mathbf{a}}$ is given by

$$\hat{\mathbf{a}} = \frac{\mathbf{a}}{\|\mathbf{a}\|}.$$

Differential Identities of Euclidean Norm

The gradient of the Euclidean norm of a vector, with respect to the vector of the norm, is the corresponding unit vector,

$$\nabla_{\mathbf{a}}||\mathbf{a}|| = \hat{\mathbf{a}}. \tag{C.10}$$

The proof, which is omitted, involves substituting s for $(\mathbf{a}_x^2 + \mathbf{a}_y^2 + \mathbf{a}_z^2)$, applying the chain rule, and then solving the partial differential of s with respect to each of $\mathbf{a}_x$, $\mathbf{a}_y$, and $\mathbf{a}_z$.

Suppose the vector $\mathbf{a}$ is a vector separating any two points $\mathbf{b}$ and $\mathbf{c}$. What then is the gradient of the Euclidean norm of $\mathbf{a}$, as the end points $\mathbf{b}$ and $\mathbf{c}$ are varied respectively? If we define $\mathbf{a} = \mathbf{c} - \mathbf{b}$, then the gradient of $||\mathbf{a}||$, while varying $\mathbf{c}$'s position and keeping $\mathbf{b}$ constant, is

$$\nabla_{\mathbf{c}}||\mathbf{a}|| = \hat{\mathbf{a}}. \tag{C.11}$$

And the gradient while keeping $\mathbf{c}$ constant and varying $\mathbf{b}$'s position is

$$\nabla_{\mathbf{b}}||\mathbf{a}|| = -\hat{\mathbf{a}}. \tag{C.12}$$

The proofs are similar to the proof of (C.10) except that as each partial derivative is evaluated, then the chain rule is applied a second time. Again, proofs are omitted.

Gradient of Particle Separation Distance

The potential energy functions we use are defined in part as a function of particle separation, so we now derive equations describing the gradient of the separation distance with respect to change in position and to change in orientation. Let us assume we have two particles $\mathbf{x}_i$ and $\mathbf{x}_j$, and a separation distance vector defined as $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$. Applying the general identities (C.11) and (C.12), we arrive at the following equations for the gradient with respect to the change in positions of $\mathbf{x}_i$ and $\mathbf{x}_j$,

$$\nabla_{\mathbf{x}_i}||\mathbf{r}_{ij}|| = -\hat{\mathbf{r}}_{ij}, \tag{C.13}$$
$$\nabla_{\mathbf{x}_j}||\mathbf{r}_{ij}|| = \hat{\mathbf{r}}_{ij}. \tag{C.14}$$

In deriving the torque equations we will need to evaluate the gradient with respect to the particle normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$. Since $\mathbf{r}_{ij}$ is not a function of either normal vector, we have,

$$\nabla_{\mathbf{n}_i}||\mathbf{r}_{ij}|| = 0, \tag{C.15}$$
$$\nabla_{\mathbf{n}_j}||\mathbf{r}_{ij}|| = 0. \tag{C.16}$$

Gradient of the Difference of Normal Vectors

For the case of the co-normality potential we consider the gradient of the difference between two normal vectors $\mathbf{n}_i$ and $\mathbf{n}_j$. Applying the identities (C.11) and (C.12) results in the following equations,

$$\nabla_{\mathbf{n}_i} \|\mathbf{n}_i - \mathbf{n}_j\| = \frac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\mathbf{n}_i - \mathbf{n}_j} \tag{C.17}$$

$$\nabla_{\mathbf{n}_j} \|\mathbf{n}_i - \mathbf{n}_j\| = -\frac{\|\mathbf{n}_i - \mathbf{n}_j\|}{\mathbf{n}_i - \mathbf{n}_j}. \tag{C.18}$$

And since the Euclidean norm is constant as the particle positions change,

$$\nabla_{\mathbf{x}_i} \|\mathbf{n}_i - \mathbf{n}_j\| = 0, \tag{C.19}$$

$$\nabla_{\mathbf{x}_j} \|\mathbf{n}_i - \mathbf{n}_j\| = 0. \tag{C.20}$$

C.4 Variations of Scalar Products

The scalar product of any two vectors $\mathbf{a}$ and $\mathbf{b}$ is defined to be

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{a}_x \mathbf{b}_x + \mathbf{a}_y \mathbf{b}_y + \mathbf{a}_z \mathbf{b}_z.$$

Differential Identities of Scalar Products

For any two vectors $\mathbf{a}$ and $\mathbf{b}$, the gradient of their scalar product is

$$\nabla_{\mathbf{a}}(\mathbf{a} \cdot \mathbf{b}) = \mathbf{b} \tag{C.21}$$

with respect to varying $\mathbf{a}$. Given any three independent vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$, by the distributive property of the scalar product and (C.21), we have

$$\nabla_{\mathbf{b}}(\mathbf{a} \cdot (\mathbf{b} - \mathbf{c})) = \mathbf{a}, \tag{C.22}$$

$$\nabla_{\mathbf{c}}(\mathbf{a} \cdot (\mathbf{b} - \mathbf{c})) = -\mathbf{a}. \tag{C.23}$$

Variations of the Scalar Product of the Normal and Separation Vectors

The co-planarity potential and the co-circularity potential functions are functions of the scalar product of normal vectors and particle separation, that is $\nabla(\mathbf{n}_i \cdot \mathbf{r}_{ij})$. To derive the gradient with respect to a particle position we apply identities (C.22) and (C.23). The relevant equations are:

$$\nabla_{\mathbf{x}_i}(\mathbf{n}_i \cdot \mathbf{r}_{ij}) = -\mathbf{n}_i, \tag{C.24}$$

$$\nabla_{\mathbf{x}_i}(\mathbf{n}_j \cdot \mathbf{r}_{ij}) = -\mathbf{n}_j, \tag{C.25}$$

$$\nabla_{\mathbf{x}_j}(\mathbf{n}_i \cdot \mathbf{r}_{ij}) = \mathbf{n}_i, \tag{C.26}$$

$$\nabla_{\mathbf{x}_j}(\mathbf{n}_j \cdot \mathbf{r}_{ij}) = \mathbf{n}_j. \tag{C.27}$$

To derive the gradient of the product with respect to a given particle normal, we apply identity (C.21). The relevant equations are:

$$\begin{aligned}
\nabla_{\mathbf{n}_i}(\mathbf{n}_i \cdot \mathbf{r}_{ij}) &= \mathbf{r}_{ij}, && \text{(C.28)}\\
\nabla_{\mathbf{n}_i}(\mathbf{n}_j \cdot \mathbf{r}_{ij}) &= 0, && \text{(C.29)}\\
\nabla_{\mathbf{n}_j}(\mathbf{n}_i \cdot \mathbf{r}_{ij}) &= 0, && \text{(C.30)}\\
\nabla_{\mathbf{n}_j}(\mathbf{n}_j \cdot \mathbf{r}_{ij}) &= \mathbf{r}_{ij}. && \text{(C.31)}
\end{aligned}$$

C.5 Variation of the Weighting Function

The scalar weighting function $w(\|\mathbf{r}_{ij}\|)$ is a monotonically decreasing function used to limit the range of inter-particle interactions, where $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$ is defined as the distance vector between points $\mathbf{x}_i$ and $\mathbf{x}_j$. In evaluating the gradients of the weighting function we apply the chain rule (C.4) and equations (C.13), (C.14), (C.15), and (C.16). Using the following shorthand notation,

$$\begin{aligned}
w &= w(\|\mathbf{r}_{ij}\|),\\
w' &= \frac{\mathrm{d}w(\|\mathbf{r}_{ij}\|)}{\mathrm{d}\|\mathbf{r}_{ij}\|}
\end{aligned}$$

we have,

$$\begin{aligned}
\nabla_{\mathbf{x}_i} w &= w'\nabla_{\mathbf{x}_i}\|\mathbf{r}_{ij}\| = -\hat{\mathbf{r}}_{ij} w', && \text{(C.32)}\\
\nabla_{\mathbf{x}_j} w &= w'\nabla_{\mathbf{x}_j}\|\mathbf{r}_{ij}\| = \hat{\mathbf{r}}_{ij} w', && \text{(C.33)}\\
\nabla_{\mathbf{n}_i} w &= w'\nabla_{\mathbf{n}_i}\|\mathbf{r}_{ij}\| = 0, && \text{(C.34)}\\
\nabla_{\mathbf{n}_j} w &= w'\nabla_{\mathbf{n}_j}\|\mathbf{r}_{ij}\| = 0. && \text{(C.35)}
\end{aligned}$$

C.6 Forces and Torques from Weighted Potentials

In this section we use the results from the previous sections to show how to derive force and torque vectors from weighted scalar potential fields.

Force from a Weighted Potential

The gradient of a weighted scalar potential is computed based on the application of the multiplication and chain rules. Assuming we have two particles positioned at $\mathbf{x}_i$ and $\mathbf{x}_j$, the vector $\mathbf{r}_{ij}$ separating the two particles is $(\mathbf{x}_j - \mathbf{x}_i)$ and the separation distance is $r_{ij} = \|\mathbf{r}_{ij}\|$. Given a potential $\phi(r)$ and a weighting $w(r)$, both scalar functions of distance, the gradient of their product is

$$\nabla_{\mathbf{x}}\left(w(r_{ij})\phi(r_{ij})\right) = w(r_{ij})\frac{\mathrm{d}\phi(r_{ij})}{\mathrm{d}r}\nabla_{\mathbf{x}} r_{ij} + \frac{\mathrm{d}w(r_{ij})}{\mathrm{d}r}\phi(r_{ij})\nabla_{\mathbf{x}} r_{ij}.$$

Applying (C.1), (C.13) and (C.14) the resultant force acting on particle i is thus

$$\mathbf{f}_i = -\nabla_{\mathbf{x}_i}\left(w(r)\phi(r)\right) = \hat{\mathbf{r}}_{ij}\left(w(r_{ij})\frac{\mathrm{d}\phi(r_{ij})}{\mathrm{d}r} + \phi(r_{ij})\frac{\mathrm{d}w(r_{ij})}{\mathrm{d}r}\right)$$

and the force acting on particle j is

$$\mathbf{f}_j = -\nabla_{\mathbf{x}_j}\left(w(r)\phi(r)\right) = -\mathbf{f}_i.$$

It should be clear that these equations also hold for potentials that are a function not only of particle separation but also of particle normals.

Torque from a Weighted Potential

Assuming we have two particles as above and with normals given by $\mathbf{n}_i$ and $\mathbf{n}_j$, and given a potential $\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)$ and a weighting $w(r_{ij})$, then the gradient of their product with respect to variation in normal is

$$\nabla_{\mathbf{n}}\left(w(r_{ij})\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right) = w(r_{ij})\nabla_{\mathbf{n}}\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j) + \phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\nabla_{\mathbf{n}}w(r_{ij}).$$

And since the gradient of the weighting function is zero this reduces to

$$\nabla_{\mathbf{n}}\left(w(r_{ij})\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right) = w(r_{ij})\nabla_{\mathbf{n}}\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j).$$

By (C.2), (C.9), and (C.6) the torque vector acting on particle i is

$$\boldsymbol{\tau}_i = -\nabla_{\boldsymbol{\theta}}\left(w(r_{ij})\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right) = -\mathbf{n}_i \times \left(\nabla_{\mathbf{n}_i}\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right) w(r_{ij}).$$

Likewise the torque acting on particle j is

$$\boldsymbol{\tau}_i = -\mathbf{n}_j \times \left(\nabla_{\mathbf{n}_j}\phi(r_{ij}, \mathbf{n}_i, \mathbf{n}_j)\right) w(r_{ij}).$$

Appendix D

Computation of internal forces

Based on the identities and equations of Appendix C, we derive the inter-particle forces and torques for the distance weighted versions of the co-planarity, the co-normality, the co-circularity, and the Lennard-Jones potentials. For a pair of particles i and j we derive the forces and torques with respect to both particle i and particle j.

D.1 The Co-planarity Potential

The spatially weighted co-planarity potential is

$$\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 w.$$

To derive the forces and torques, we first derive the gradients of the potential function with respect to the variation in particle positions, particle normals, and the infinitesimal change in particle orientations.

Variation in Position

To derive the gradient with respect to the variation in the particle i's position we apply the multiplication rule (C.3) resulting in two parts to differentiate. To the left side we apply equation (C.32), and to the right side we apply the chain rule (C.4) and equation (C.24):

$$\nabla_{\mathbf{x}_i}\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \nabla_{\mathbf{x}_i} w + w \nabla_{\mathbf{x}_i} (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 w'(-\hat{\mathbf{r}}_{ij}) + w2(\mathbf{n}_i \cdot \mathbf{r}_{ij})(-\mathbf{n}_i).$$

The derivation of the gradient with respect to particle j's position is similar, except we apply equations (C.33) and (C.26) where appropriate:

$$\nabla_{\mathbf{x}_j}\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \nabla_{\mathbf{x}_j} w + w \nabla_{\mathbf{x}_j} (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 w' \hat{\mathbf{r}}_{ij} + w2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i.$$

Variation in Normal

To derive the gradient with respect to the variation in particle i's normal we apply the multiplication rule (C.3), the chain rule (C.4), and equations (C.34) and (C.28):

$$\nabla_{\mathbf{n}_i}\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \nabla_{\mathbf{n}_i} w + w \nabla_{\mathbf{n}_i} (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 = 2w(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{r}_{ij}.$$

The derivation with respect to particle j's normal is similar, except we apply equations (C.35) and (C.30) where appropriate:

$$\nabla_{\mathbf{n}_j}\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \nabla_{\mathbf{n}_j} w + w \nabla_{\mathbf{n}_j} (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 = 0.$$

Variation in Orientation

To derive the gradient with respect to the infinitesimal change in the particle orientation we apply equation (C.9) and the above results:

$$\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{P}} = \mathbf{n}_i \times \nabla_{\mathbf{n}_i}\phi_{\mathrm{P}} = 2w(\mathbf{n}_i \cdot \mathbf{r}_{ij})(\mathbf{n}_i \times \mathbf{r}_{ij})w.$$

The gradient of the potential with respect to the change in orientation of particle j is zero, because the potential is independent of the orientation of particle j:

$$\nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{P}} = \mathbf{n}_j \times \nabla_{\mathbf{n}_j}\phi_{\mathrm{P}} = \mathbf{n}_j \times 0 = 0.$$

Forces and Torques

The forces and torques follow directly from the above derivations:

$$\begin{aligned}
\mathbf{f}_{\mathrm{P}_i} &= -\nabla_{\mathbf{x}_i}\phi_{\mathrm{P}} = (\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \hat{\mathbf{r}}_{ij} w' + 2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i w \\
\mathbf{f}_{\mathrm{P}_j} &= -\nabla_{\mathbf{x}_j}\phi_{\mathrm{P}} = -(\mathbf{n}_i \cdot \mathbf{r}_{ij})^2 \hat{\mathbf{r}}_{ij} w' - 2(\mathbf{n}_i \cdot \mathbf{r}_{ij})\mathbf{n}_i w = -\mathbf{f}_{\mathrm{P}_i} \\
\boldsymbol{\tau}_{\mathrm{P}_i} &= -\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{P}} = -2(\mathbf{n}_i \cdot \mathbf{r}_{ij})(\mathbf{n}_i \times \mathbf{r}_{ij})w \\
\boldsymbol{\tau}_{\mathrm{P}_j} &= -\nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{P}} = 0.
\end{aligned}$$

D.2 The Co-normality Potential

The spatially weighted co-normality potential is

$$\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 w.$$

To derive the forces and torques, we first derive the gradients of the potential function with respect to the variation in particle positions, particle normals, and the infinitesimal change in particle orientations.

Variation in Position

To derive the gradient with respect to the variation in the particle i's position, we apply the multiplication rule (C.3) resulting in a left and right side to differentiate. To the left side we apply equation (C.32), and to the right side we apply the chain rule (C.4) and equation (C.19):

$$\nabla_{\mathbf{x}_i}\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \nabla_{\mathbf{x}_i} w + w \nabla_{\mathbf{x}_i}\left(\|\mathbf{n}_i - \mathbf{n}_j\|^2\right) = \|\mathbf{n}_i - \mathbf{n}_j\|^2(-\mathbf{r}_{ij})w'.$$

The derivation of the gradient with respect to particle j's position is similar, except we apply equations (C.33) and (C.20) where appropriate:

$$\nabla_{\mathbf{x}_j}\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \nabla_{\mathbf{x}_j} w + w \nabla_{\mathbf{x}_j}\left(\|\mathbf{n}_i - \mathbf{n}_j\|^2\right) = \|\mathbf{n}_i - \mathbf{n}_j\|^2\mathbf{r}_{ij}w'.$$

Variation in Normal

To derive the gradient with respect to the variation in particle i's normal we apply the multiplication rule (C.3), the chain rule (C.4), and equations (C.34) and (C.17):

$$\nabla_{\mathbf{n}_i}\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \nabla_{\mathbf{n}_i} w + w \nabla_{\mathbf{n}_i}\left(\|\mathbf{n}_i - \mathbf{n}_j\|^2\right) = w2(\mathbf{n}_i - \mathbf{n}_j).$$

The derivation with respect to particle j's normal is similar, except we apply equations (C.35) and (C.18) where appropriate:

$$\nabla_{\mathbf{n}_j}\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2 \nabla_{\mathbf{n}_j} w + w \nabla_{\mathbf{n}_j}\left(\|\mathbf{n}_i - \mathbf{n}_j\|^2\right) = -w2(\mathbf{n}_i - \mathbf{n}_j).$$

Variation in Orientation

To derive the gradient with respect to the infinitesimal change in the particle orientation we apply equation (C.9) and the above results:

$$\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{N}} = \mathbf{n}_i \times \nabla_{\mathbf{n}_i}\phi_{\mathrm{N}} = 2w\left(\mathbf{n}_j \times \mathbf{n}_i\right).$$

The gradient with respect to the infinitesimal change in particle j's orientation is derived by a similar process :

$$\nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{N}} = \mathbf{n}_j \times \nabla_{\mathbf{n}_j}\phi_{\mathrm{N}} = -2w\left(\mathbf{n}_j \times \mathbf{n}_i\right).$$

Forces and Torques

The forces and torques follow directly from the above derivations:

$$\begin{aligned}
\mathbf{f}_{\mathrm{N}_i} &= -\nabla_{\mathbf{x}_i}\phi_{\mathrm{N}} = \|\mathbf{n}_i - \mathbf{n}_j\|^2\mathbf{r}_{ij}w' \\
\mathbf{f}_{\mathrm{N}_j} &= -\nabla_{\mathbf{x}_j}\phi_{\mathrm{N}} = -\|\mathbf{n}_i - \mathbf{n}_j\|^2\mathbf{r}_{ij}w' = -\mathbf{f}_{\mathrm{N}_i} \\
\boldsymbol{\tau}_{\mathrm{N}_i} &= -\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{N}} = -2\left(\mathbf{n}_j \times \mathbf{n}_i\right)w \\
\boldsymbol{\tau}_{\mathrm{N}_j} &= -\nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{N}} = 2\left(\mathbf{n}_j \times \mathbf{n}_i\right)w = -\boldsymbol{\tau}_{\mathrm{N}_i}.
\end{aligned}$$

D.3 The Co-circularity Potential

The spatially weighted co-circularity potential is

$$\phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 w.$$

To derive the forces and torques, we first derive the gradients of the potential function with respect to the variation in particle positions, particle normals, and the infinitesimal change in particle orientations.

Variation in Position

To derive the gradient with respect to the variation in the particle i's position, we apply the multiplication rule (C.3) resulting in a left and right side to differentiate. To the left side we apply equation (C.32), while we apply the chain rule (C.4) to the right side:

$$\begin{aligned}
\nabla_{\mathbf{x}_i}\phi_{\mathrm{C}} &= \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \nabla_{\mathbf{x}_i} w + w \nabla_{\mathbf{x}_i} \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \\
&= \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 (-\hat{\mathbf{r}}_{ij}) w' + w 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right) \nabla_{\mathbf{x}_i} \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right).
\end{aligned}$$

To solve the gradient on right hand side of the equation we apply the distributive rule of scalar products, the identity $\nabla(\mathbf{a} + \mathbf{b}) = (\nabla \mathbf{a}) + (\nabla \mathbf{b})$, and equations (C.24) and (C.25), thus the gradient is

$$\nabla_{\mathbf{x}_i}\phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 (-\hat{\mathbf{r}}_{ij}) w' - w 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right) (\mathbf{n}_i + \mathbf{n}_j).$$

The derivation of the gradient with respect to particle j's position is similar, except we apply equations (C.33), (C.26), and (C.27) where appropriate:

$$\nabla_{\mathbf{x}_j}\phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \hat{\mathbf{r}}_{ij} w' + w 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right) (\mathbf{n}_i + \mathbf{n}_j).$$

Variation in Normal

To derive the gradient with respect to the variation in particle i's normal we apply the multiplication rule (C.3), the chain rule (C.4), and equations (C.34), (C.28), and (C.29):

$$\nabla_{\mathbf{n}_i}\phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \nabla_{\mathbf{n}_i} w + w \nabla_{\mathbf{n}_i} \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 = w 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right) \mathbf{r}_{ij}.$$

The derivation with respect to particle j's normal is similar, except we apply equations (C.35), (C.30), and (C.31) where appropriate:

$$\nabla_{\mathbf{n}_j}\phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 \nabla_{\mathbf{n}_j} w + w \nabla_{\mathbf{n}_j} \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right)^2 = w 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij}\right) \mathbf{r}_{ij}.$$

Variation in Orientation

To derive the gradient with respect to the infinitesimal change in the particle orientation we apply equation (C.9) and the above results:

$$\nabla_{\boldsymbol{\theta}_i} \phi_{\mathrm{C}} = \mathbf{n}_i \times \nabla_{\mathbf{n}_i} \phi_{\mathrm{C}} = 2w \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_i \times \mathbf{r}_{ij}) \, .$$

The gradient with respect to the infinitesimal change in particle j's orientation is derived by a similar process:

$$\nabla_{\boldsymbol{\theta}_j} \phi_{\mathrm{C}} = \mathbf{n}_j \times \nabla_{\mathbf{n}_j} \phi_{\mathrm{C}} = 2w \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_j \times \mathbf{r}_{ij}) \, .$$

Forces and Torques

The forces and torques follow directly from the above derivations:

$$\begin{aligned}
\mathbf{f}_{\mathrm{C}_i} &= -\nabla_{\mathbf{x}_i} \phi_{\mathrm{C}} = \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right)^2 \hat{\mathbf{r}}_{ij} w' + 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_i + \mathbf{n}_j) w \\
\mathbf{f}_{\mathrm{C}_j} &= -\nabla_{\mathbf{x}_j} \phi_{\mathrm{C}} = -\left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right)^2 \hat{\mathbf{r}}_{ij} w' - 2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_i + \mathbf{n}_j) w = -\mathbf{f}_{\mathrm{C}_i} \\
\boldsymbol{\tau}_{\mathrm{C}_i} &= -\nabla_{\boldsymbol{\theta}_i} \phi_{\mathrm{C}} = -2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_i \times \mathbf{r}_{ij}) \, w \\
\boldsymbol{\tau}_{\mathrm{C}_j} &= -\nabla_{\boldsymbol{\theta}_j} \phi_{\mathrm{C}} = -2 \left((\mathbf{n}_i + \mathbf{n}_j) \cdot \mathbf{r}_{ij} \right) (\mathbf{n}_j \times \mathbf{r}_{ij}) \, w
\end{aligned}$$

D.4 The Lennard-Jones Potential

The Lennard-Jones function is a scalar function defined in three-space as a function of particle separation. Similar to the shorthand notation used for the weighting function, we will use

$$\begin{aligned}
\phi_{\mathrm{LJ}} &= \phi_{\mathrm{LJ}}(\|\mathbf{r}_{ij}\|) \\
\phi_{\mathrm{LJ}}{}' &= \frac{\mathrm{d}\phi_{\mathrm{LJ}}}{\mathrm{d}\|\mathbf{r}_{ij}\|}
\end{aligned}$$

in our derivations. To derive the forces and torques, we first derive the gradients with respect to variation in particle positions, particle normals, and the infinitesimal change in particle orientations.

Variation in Position

To derive the gradient with respect to the variation in the particle positions we apply the chain rule (C.4), and equations (C.13) and (C.14):

$$\begin{aligned}
\nabla_{\mathbf{x}_i} \phi_{\mathrm{LJ}} &= \phi_{\mathrm{LJ}}{}' \nabla_{\mathbf{x}_i} \|\mathbf{r}_{ij}\| = -\hat{\mathbf{r}}_{ij} \phi_{\mathrm{LJ}}{}' \\
\nabla_{\mathbf{x}_j} \phi_{\mathrm{LJ}} &= \phi_{\mathrm{LJ}}{}' \nabla_{\mathbf{x}_j} \|\mathbf{r}_{ij}\| = \hat{\mathbf{r}}_{ij} \phi_{\mathrm{LJ}}{}'.
\end{aligned}$$

Variation in Normal

To derive the gradient with respect to the variation in the particle normals we apply the chain rule (C.4), and equations (C.15) and (C.16):

$$\begin{aligned}\nabla_{\mathbf{n}_i}\phi_{\mathrm{LJ}} &= {\phi_{\mathrm{LJ}}}'\nabla_{\mathbf{n}_i}\|\mathbf{r}_{ij}\| = {\phi_{\mathrm{LJ}}}'(0) = 0\\ \nabla_{\mathbf{n}_j}\phi_{\mathrm{LJ}} &= {\phi_{\mathrm{LJ}}}'\nabla_{\mathbf{n}_j}\|\mathbf{r}_{ij}\| = {\phi_{\mathrm{LJ}}}'(0) = 0.\end{aligned}$$

Variation in Orientation

To derive the gradient with respect to the infinitesimal change in the particle orientation we apply equation (C.9) and the above results:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{LJ}} &= \mathbf{n}_i \times \nabla_{\mathbf{n}_i}\phi_{\mathrm{LJ}} = \mathbf{n}_i \times 0 = 0\\ \nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{LJ}} &= \mathbf{n}_j \times \nabla_{\mathbf{n}_j}\phi_{\mathrm{LJ}} = \mathbf{n}_j \times 0 = 0.\end{aligned}$$

Forces and Torques

The forces and torques follow directly from the above derivations:

$$\begin{aligned}\mathbf{f}_{\mathrm{LJ}_i} &= -\nabla_{\mathbf{x}_i}\phi_{\mathrm{LJ}} = \hat{\mathbf{r}}_{ij}{\phi_{\mathrm{LJ}}}'\\ \mathbf{f}_{\mathrm{LJ}_j} &= -\nabla_{\mathbf{x}_j}\phi_{\mathrm{LJ}} = -\hat{\mathbf{r}}_{ij}{\phi_{\mathrm{LJ}}}' = -\mathbf{f}_j\\ \boldsymbol{\tau}_{\mathrm{LJ}_i} &= -\nabla_{\boldsymbol{\theta}_i}\phi_{\mathrm{LJ}} = 0\\ \boldsymbol{\tau}_{\mathrm{LJ}_j} &= -\nabla_{\boldsymbol{\theta}_j}\phi_{\mathrm{LJ}} = 0\end{aligned}$$

Appendix E

Finite Element Analysis of Surface 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

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

and the average variation in curvature from

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

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$^{\mathrm{TM}}$(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, using Mathematica™, we obtain

$$V = h^{-2} 6\sqrt{3} p_+^2, \tag{E.1}$$

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

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

To 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

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

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 use the local z direction along the average normal vector $(\mathbf{n}_i + \mathbf{n}_j)/2$, which leads to the equations

$$q_+ = -(\frac{\mathbf{n}_i + \mathbf{n}_j}{2}) \cdot \widehat{\frac{\mathbf{n}_i + \mathbf{n}_j}{2}} \times \hat{\mathbf{r}}_{\mathbf{ij}} = \mathbf{0}, \tag{E.4}$$

$$q_- = -(\frac{\mathbf{n}_i - \mathbf{n}_j}{2}) \cdot \widehat{\frac{\mathbf{n}_i + \mathbf{n}_j}{2}} \times \hat{\mathbf{r}}_{\mathbf{ij}}, \tag{E.5}$$

$$p_-^2 + q_-^2 \approx \frac{1}{4}\|\mathbf{n}_i - \mathbf{n}_j\|^2. \tag{E.6}$$

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 (E.2) corresponds directly to the co-circularity ϕ_{C} (4.17). The curvature itself C (E.3) can be written as a sum of the co-circularity potential and the co-normality potential ϕ_{N} (4.14). 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 (E.3) is too complicated to warrant direct implementation, finite rest area behavior is simulated by the Lennard-Jones interaction potential ϕ_{LJ}.

References

Agin, G. J. and Binford, T. O. (1973). Computer description of curved objects. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 629–640. IJCAI.

Bajaj, C. L. and Ihm, I. (1992). Smoothing polyhedra using implicit algebraic splines. *Computer Graphics (Proceedings SIGGRAPH'92)*, 26(2):79–88.

Baker, H. H. (1988). Building, visualizing, and computing on surfaces of evolution. *IEEE Computer Graphics and Applications*, 8(4):31–41.

Baraff, D. (1991). Rigid body concepts. In *SIGGRAPH '91 Course notes #23: An Introduction to Physically Based Modeling*. SIGGRAPH.

Barr, A. H. (1981). Superquadrics and angle preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23.

Barr, A. H. (1984). Global and local deformations of solid primitives. *Computer Graphics (SIGGRAPH'84)*, 18(3):21–30.

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

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

Barton, A. (1974). *The Dynamic Liquid State*. Longman Group Limited, New York.

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

Bloomenthal, J. (1988). Polygonization of implicit surfaces. *Computer Aided Geometric Design*, 5(4):341–355.

Bloomenthal, J. (1989). Techniques for implicit modeling. Technical Report P89-00106, Xerox PARC.

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

Boult, T. E. and Kender, J. R. (1986). Visual surface reconstruction using sparse depth data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'86)*, pages 68–76, Miami Beach, Florida.

Breen, D., House, D., and Getto, P. (1991). A particle-based computational model of cloth draping. In *Scientific Visualization of Physical Phenomena (Proceedings of Computer Graphics International '91)*, pages 113–134. Springer-Verlag.

Breen, D. E., House, D. H., and Wozny, M. J. (1994). Predicting the drape of woven cloth using interacting particles. *Computer Graphics (Proceedings SIGGRAPH'94)*, 28(3):365–372.

Carlbom, I., Hsu, W. M., Klinker, G., Szeliski, R., Waters, K., Doyle, M., Gettys, J., Harris, K. M., Levergood, T. M., Palmer, R., Palmer, L., Picart, M., Terzopoulos, D., Tonnesen, D., Vannier, M., and Wallace, G. (1992). Modeling and analysis of empirical data in collaborative environments. *Communications of the ACM*, 35(6):75–84.

Case, J. (1938). *The Strength of Materials*. Edward Arnold and Co., London, third edition.

Catmull, E. and Clark, J. (1978). Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-aided Design*, 10(6):350–355.

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

Chen, J. X. and Lobo, N. V. (1995). Toward interactive-rate simulation of fluids with moving obstacles using navier-stokes equations. *Graphical Models and Image Processing*, 57(2):107–116.

Chew, L. P. (1993). Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ACM Symposium on Computational Geometry*, pages 274–280.

Christy, R. W. and Pytte, A. (1965). *The Structure of Matter: An Introduction to Modern Physics*. W. A. Benjamin, New York.

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

Crossno, P. and Angel, E. (1997). Isosurface extraction using particle systems. In *Proceedings of the conference on Visualization '97*.

de Berg, M., van Kreveld, M., Overmars, M., and Schwarzkopf, O. (1997). *Computational Geometry Algorithms and Applications*. Springer-Verlag, Berlin.

Desbrun, M. and Gascuel, M. (1995). Animating soft substances with implicit surfaces. *Computer Graphics (Proceedings SIGGRAPH'95)*, pages 287–290.

Desbrun, M. and Gascuel, M. (1996). Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Animation and Simulation '96*.

Doo, D. and Sabin, M. (1978). Behaviour of recursive division surfaces near extraordinary points. *Computer-aided Design*, 10(6):356–360.

Dwyer, R. (1991). Higher-dimensional voronoi diagrams in linear expected time. *Discrete Computational Geometry*, pages 343–367.

Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72.

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

Fleischer, K., Laidlaw, D., Currin, D., and Barr, A. (1995). Cellular texture generation. *Computer Graphics (Proceedings SIGGRAPH'95)*, pages 239–248.

Foley, J. D., Van Dam, A., Feiner, S. K., and Hughes, J. F. (1990). *Computer Graphics: Principles and Practice*. Addison-Wesley, 2nd edition.

Forsey, D. R. and Bartels, R. H. (1988). Hierarchical b-spline refinement. *Computer Graphics (Proceedings SIGGRAPH'88)*, 22(4):205–212.

Foster, N. and Metaxas, D. (1996). Realistic animation of liquids. In *Graphics Interface*, pages 204–212.

Foster, N. and Metaxas, D. (1997a). Controlling fluid animation. In *Graphics Interface*, pages 178–188.

Foster, N. and Metaxas, D. (1997b). Modeling the motion of a hot, turbulent gas. *Computer Graphics (Proceedings SIGGRAPH'97)*.

Fournier, A. and Reeves, W. T. (1986). A simple model of ocean waves. *Computer Graphics (SIGGRAPH'86)*, 20(4):75–84.

Franke, R. (1982). Scattered data interpolation: Tests of some methods. *Mathematics of Computation*, 38(157):181–200.

Fua, P. and Sander, P. (1992). Reconstructing surfaces from unstructured 3d points. In *Second European Conference on Computer Vision (ECCV'92)*, pages 50–55, Sta. Margherita, Italy. Springer-Verlag.

Gasson, P. C. (1983). *Geometry of Spatial Forms*. Ellis Horwood Limited.

Glassner, A., editor (1993). *Graphic Gems I*. Academic Press.

Goldstein, H. (1950). *Classical Mechanics*. Addison-Wesley Publishing Co.

Goodman, J. and O'Rourke, J., editors (1997). *Handbook of Discrete and Computational Geometry*. CRC Press, New York.

Goss, M. E. (1990). A real time particle system for display of ship wakes. *IEEE Computer Graphics and Applications*, 10(3):30–35.

Gould, H. and Tobochnik, J. (1988). *An Introduction to Computer Simulation Methods*. Addison-Wesley, Reading Massachusetts.

Gray, A. (1993). *Modern differential geometry of curves and surfaces.* CRC Press, Boca Raton, Florida.

Greengard, L. (1988). *The Rapid Evaluation of Potential Fields in Particle Systems.* The MIT Press, Boston, Massachusetts.

Greenspan, D. (1973). *Discrete Models.* Addison-Wesley, Reading, Massachusetts.

Guo, B. and Menon, J. (1996). Local shape control of free-form solids in exact csg representation. *Computer-Aided Design*, 28(6-7):483–493.

Guo, B., Menon, J., and Willette, B. (1997). Surface reconstruction using alpha shapes. *Computer Graphics Forum*, 16(4):177–190.

Haber, R. B. and McNabb, D. A. (1990). Visualization idioms: A conceptual model for scientific visualization systems. In *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press.

Hall, M. and Warren, J. (1990). Adaptive polygonalization of implicitly defined surfaces. *IEEE Computer Graphics and Applications*, 10(6):33–42.

Haumann, D., Wejchert, J., Arya, K., Bacon, B., Khorasani, A., Norton, A., and Sweeney, P. (1991). An application of motion design and control for physically-based animation. In *Proceedings of Graphics Interface '91*, pages 279–286.

Hay, G. E. (1953). *Vector and Tensor Analysis.* Dover Publications, Inc.

Heggie, C. (1987). The n-body problem in stellar dynamics. In Roy, A. E., editor, *NATA Advanced Studies Institute: a long-term Dynamical Behaviour of Natural and Artificial N-Body Systems*, pages 329–347. Kluwer Academic Publishers, Boston.

Heyes, D. M. (1998). *The liquid state: applications of molecular simulations.* John Wiley and Sons Ltd.

Hibbard, W. and Santek, D. (1989). Interactivity is the key. In *Chapel Hill Workshop on Volume Visualization, Conference Proceedings*, pages 39–43, Chapel Hill, North Carolina.

Higgins, W. E., Chung, N., and Ritman, E. L. (1990). Extraction of left-ventricular chamber from 3-D CT images of the heart. *IEEE Transactions on Medical Imaging*, 9(4).

Hockney, R. W. and Eastwood, J. W. (1981). *Computer Simulation using Particles.* McGraw-Hill Inc., New York.

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

Hoffman, C. M. (1989). *Geometric and Solid Modeling.* Morgan Kaufmann Publishers.

Hoffmann, B. (1966). *About Vectors.* Dover Publications, New York.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Computer Graphics (Proceedings SIGGRAPH'92)*, 26(2):71–78.

Houwink, R. and de Decker, H. K., editors (1971). *Elasticity, Plasticity, & Structure of Matter.* Cambridge University Press, 3rd edition.

Hsu, W. M., Hughes, J. F., and Kaufman, H. (1992). Direct manipulation of free-form deformations. *Computer Graphics (Proceedings SIGGRAPH '92)*, 26(2):177–184.

Jaeger, J. C. (1969). *Elasticity, Fracture and Flow.* Butler and Tanner Ltd, London, 3rd edition.

Kass, M. and Miller, G. (1990). Rapid, stable fluid dynamics for computer graphics. *Computer Graphics (SIGGRAPH'90)*, 24(4):49–55.

Kass, M., Witkin, A., and Terzopoulos, D. (1987). Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331.

Koenderink, J. J. (1990). *Solid Shape.* MIT Press, Cambridge, MA.

Kreyszig, E. (1959). *Differential Geometry.* University of Toronto Press.

Lawson, C. L. (1977). Software for C^1 surface interpolation. In *Mathematical Software III (Proceedings of the Symposium on Mathematical Software '77, Madison, Wisconsin)*, pages 161–194.

Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic modeling for facial animation. *Computer Graphics (Proceedings SIGGRAPH'95)*, pages 55–62.

Lombardo, J. and Puech, C. (1995). Oriented particles: A tool for shape memory objects modelling. In *Graphics Interface*, pages 255–262.

Loop, C. (1994). A G^1 triangular spline surface of arbitrary topological type. *Computer Aided Geometric Design*, 11(3):303–330.

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

Loop, C. T. (1987). Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah.

Lord, E. A. and Wilson, C. B. (1984). *The Mathematicl Description of Shape and Form.* Ellis Horwood Ltd.

Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3-D surface construction algorithm. *Computer Graphics (SIGGRAPH'87)*, 21(4):163–169.

Madsen, K. H. (1994). A guide to metaphorical design. *Communications of the ACM*, 37(12):57–62.

Marion, J. B. (1970). *Classical Dynamics of Particles and Systems.* Academic Press, 2nd edition.

McInerney, T. and Terzopoulos, D. (1993). A finite element model for 3D shape reconstruction and nonrigid motion tracking. In *Fourth International Conference on Computer Vision (ICCV'93)*, pages 518–523. IEEE Computer Society Press.

McInerney, T. and Terzopoulos, D. (1996). Deformable models in medical image analysis: A survey. *Medical Image Analysis*, 1(2):91–108.

McInerney, T. and Terzopoulos, D. (1997). Medical image segmentation using topologically adaptable surfaces. In *Proceedings of CVRMed*, page unknown.

Meier, B. (1996). Painterly rendering for animation. *Computer Graphics (SIGGRAPH'96)*, pages 477–484.

Miller, G. and Pearce, A. (1989). 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.

Miller, G. S. P. (1988). The motion dynamics of snakes and worms. *Computer Graphics (Proceedings SIGGRAPH'88)*, 22(4):169–178.

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

Miner, D. F. and Seastone, J. B., editors (1955). *Handbook of Engineering Materials.* John Wiley and Sons.

Monaghan, J. J. (1982). Why particle methods work. *SIAM J. Sci. Stat. Computing*, 3(4):422–433.

Monaghan, J. J. (1985). Particle methods for hydrodynamics. *Computer Physics Reports*, 3(2):71–124.

Monaghan, J. J. (1988). An introduction to SPH. *Computer Physics Communications*, 48(1):543–574.

Monaghan, J. J. (1992). Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574.

Monaghan, J. J., Thompson, M. C., and Hourigan, K. (1994). Simulation of free surface flows with SPH. ASME Symposium on Computational Methods in Fluid Dynamics. http://www-personal.monash.edu.au/ mecheng/mct/docs/sph/sph.html (Mar. 1998).

Monga, O., Deriche, R., Maladin, G., and Cocquerez, J. P. (1990). Recursive filtering and edge closing: two primary tools for 3D edge detection. In *First European Conference on Computer Vision (ECCV'90)*, pages 56–65, Antibes, France. Springer-Verlag.

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

Mould, D. and Yang, Y.-H. (1997). Modeling water for computer graphics. *Computer and Graphics*, 21(6):801–814.

Nevatia, R. and Binford, T. O. (1977). Description and recognition of curved objects. *Artificial Intelligence*, 8:77–98.

Ney, D., Fishman, E. K., and Magid, D. (1990). Three dimensional imaging of computed tomography: Techniques and applications. In *First Conference on Visualization in Biomedical Computing (VBC'90)*, pages 498–506, Los Alamitos, California. IEEE Computer Society Press.

O'Brien, J. F. and Hodgins, J. K. (1994). Dynamic simulation of splashing fluids. Technical Report GIT-GVU-94-32, Georgia Institute of Technology.

O'Brien, J. F. and Hodgins, J. K. (1995). Dynamic simulation of splashing fluids. In *Proceedings of Computer Animation '95*, pages 198–205. Geneva Switzerland, April 19-21.

Overmars, M. H. (1983). *The Design of Dynamic Data Structures.* Lecture Notes in Computer Science #156. Springer-Verlag.

Peachy, D. R. (1986). Modeling waves and surf. *Computer Graphics (SIGGRAPH'86)*, 20(4):65–74.

Pentland, A. P. (1986). Perceptual organization and the representation of natural form. *Aritificial Intelligence*, 28(3):293–331.

Peters, J. and Reif, U. (1997). The simplest subdivision scheme for smoothing polyhedra. *ACM Transactions on Graphics*, 16(4):420–431.

Preparata, F. P. and Shamos, M. (1985). *Computational Geometry - An Introduction.* Springer-Verlag, New York, NY.

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

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, second edition.

Qin, H. and Terzopoulos, D. (1996). D-nurbs: A physics-based framework for geometric design. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):85–96.

Reeves, W. T. (1983a). Particle systems — a technique for modeling a class of fuzzy objects. *Computer Graphics*, 17(3):359–376.

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

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

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (SIGGRAPH'87)*, 21(4):25–34.

Reynolds, H. (1997). An alternative inter-particle force model for coupled system flexible body dynamics. In *Proceedings of Eurographics Workshop in Budapest, Hungary '97*, pages 99–110.

Rippa, S. (1990). Minimal roughness property of the delauney triangulation. *Computer Aided Geometric Design*, 7(6):489–497.

Roy, T. M. (1995). Physically based fluid model using smoothed particle hydrodynamics. Master's thesis, University of Illinois at Chicago.

Sachs, E., Roberts, A., and Stoops, D. (1991). 3-Draw: A tool for designing 3D shapes. *IEEE Computer Graphics and Applications*, 11(6):18–26.

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

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

Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. *Computer Graphics (SIGGRAPH'86)*, 20(4):151–160.

Shoemake, K. (1985). Animating rotation with quaternion curves. *Computer Graphics (SIGGRAPH'85)*, 19(3):245–254.

Shoemake, K. (1989). Quaternion calculus for animation. In *SIGGRAPH '89, Course notes #23: Math for SIGGRAPH*, pages 187 – 205. SIGGRAPH.

Shoemake, K. (1991). Quaternions and 4x4 matrices. In *Graphic Gems II*, chapter VII.6, pages 351–354. Academic Press.

Sims, K. (1988). Particle dreams. In *SIGGRAPH Video Review*, New York. ACM SIGGRAPH.

Sims, K. (1989). Particle animation and rendering using data parallel computation. In *SIGGRAPH '89, Course 30 notes: Topics in Physically-based Modeling*, pages T1– R15, New York. SIGGRAPH.

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

Sims, K. (1992). Choreographed image flow. *The Journal Of Visualization And Computer Animation*, 3:31–43.

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

Stam, J. (1995). *Multi-Scale Stochastic Modelling of Complex Natural Phenomena.* PhD thesis, Dept. of Computer Science, University of Toronto.

Stam, J. and Fiume, E. (1993). Turbulent wind fields for gaseous phenomena. *Computer Graphics (Proceedings of SIGGRAPH'93)*, pages 369–376.

Stam, J. and Fiume, E. (1995). Depiction of fire and other gaseous phenomena using diffusion processes. *Computer Graphics (Proceedings of SIGGRAPH'95)*, pages 129–136.

Stytz, M. R., Frieder, G., and Frieder, O. (1991). Three-dimensional medical imaging: Algorithms and computer systems. *ACM Computing Surveys*, 23(4):421–499.

Szeliski, R. (1990). Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528.

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

Szeliski, R. (1993). Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32.

Szeliski, R. (1998). Private communication. May 1998.

Szeliski, R. and Tonnesen, D. (1992). Surface modeling with oriented particles. *Computer Graphics (SIGGRAPH'92)*, 26(2):185–194.

Szeliski, R., Tonnesen, D., and Terzopoulos, D. (1993a). Curvature and continuity control in particle-based surface models. In *SPIE Vol. 2031 Geometric Methods in Computer Vision II*, pages 172–181, San Diego. Society of Photo-Optical Instrumentation Engineers.

Szeliski, R., Tonnesen, D., and Terzopoulos, D. (1993b). Modeling surfaces of arbitrary topology with dynamic particles. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93)*, pages 82–87, New York, New York.

Temperley, H. N. V. (1978). *Liquids and their properties.* Ellis Horwood Limited, Chichester, England.

Terzopoulos, D. (1984). *Multiresolution Computation of Visible-Surface Representations.* PhD thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA.

Terzopoulos, D. (1986). Image analysis using multigrid relaxation methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):129–139.

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

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

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

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

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

Terzopoulos, D., Platt, J., and Fleischer, K. (1989). Heating and melting deformable models (from goop to glop). In *Proceedings Graphics Interface*, pages 219–226. Graphics Interface.

Terzopoulos, D. and Qin, H. (1994). Dynamic nurbs with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136. Special Issue on Interactive Sculpting.

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

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

Tonnesen, D. (1989). 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.

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

Trevena, D. H. (1975). *The liquid phase.* Wykeham Publications, London, England.

Tu, X. and Terzopoulos, D. (1994). Artificial fishes: Physics, locomotion, perception, behavior. *Computer Graphics (Proceedings SIGGRAPH'94)*, 28(3):43–50.

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

Vasilescu, M. and Terzopoulos, D. (1992). Adaptive meshes and shells: Irregular triangulation, discontinuities, and hierarchical subdivision. In *Computer Vision and Pattern Recognition Conference (CVPR-92)*, pages 829–832, Champaign, IL. IEEE Computer Society Press.

Velho, L. (1990). Adaptive polygonization of implicit surfaces using simplical decomposition and boundary constraints. In *Proceedings of Eurographics '90*, pages 125–136.

Vitasek, E. (1969). *Survey of Applicable Mathematics*, chapter 27: Solution of Partial Differential Equations by the Finite-Difference Method. MIT Press.

Waters, K. and Terzopoulos, D. (1990). A physical model of facial tissue and muscle articulation. In *First Conference on Visualization in Biomedical Computing (VBC'90)*, pages 77–82, Los Alamitos, California. IEEE Computer Society Press.

Welch, W. and Witkin, A. (1992). Variational surface modeling. *Computer Graphics (Proceedings SIGGRAPH'92)*, 26(2):157–166.

Welch, W. and Witkin, A. (1994). Free-form shape design using triangulated surfaces. *Computer Graphics (Proceedings SIGGRAPH'94)*, 28(3):247–256.

Witkin, A. P. and Heckbert, P. S. (1994). Using particles to sample and control implicit surfaces. *Computer Graphics (Proceedings SIGGRAPH'94)*, 28(3):269–277.

Wolfe Jr., R. H. and Liu, C. N. (1988). Interactive visualization of 3D seismic data: A volumetric method. *IEEE Computer Graphics and Applications*, 8(4):24–30.

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

Wyvill, B., McPheeters, C., and Wyvill, G. (1986a). Animating soft objects. *The Visual Computer*, 2:235–242.

Wyvill, B., McPheeters, C., and Wyvill, G. (1986b). Data structures for soft objects. *The Visual Computer*, 2:227–234.

Wyvill, G. and Trotman, A. (1989). Ray-tracing soft objects. *Proceedings of Graphics Interface 1989*, pages 469–476.

Zhai, S., Buxton, W., and Milgram, P. (1994). The “silk cursor”: Investigating transparency for 3D target acquisition. In *Proceedings of ACM CHI'94 on Human Factors in Computing Systems*, pages 459–464.

Zhao, F. (1987). An O(N) algorithm for three-dimensional N-body simulations. Master's thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, Massachusetts. Also MIT Artificial Intelligence Laboratory technical report number AI-TR-995.

Zorin, D., Schröder, P., and Sweldens, W. (1996). Interpolating subdivision for meshes with arbitrary topology. *Computer Graphics (Proceedings SIGGRAPH'96)*, pages 189–192.

Zwillinger, D., editor (1995). *CRC Standard Mathematical Tables and Formulas.* CRC Press, Inc., Baco Raton, Florida, 29th edition.