Cost Minimization for Animated Geometric Models in Computer Graphics

David E. Breen

Computer Graphics Lab
MS 348-74
California Institute of Technology
Pasadena, CA 91125

Tel (626) 395-2866
FAX (626) 793-9544
david@gg.caltech.edu

November 19, 1997

Abstract

This paper describes how the concept of imposing geometric constraints by minimizing cost functions may be utilized and extended to accomplish a variety of animated modeling tasks for computer graphics. In this approach a complex 3-D geometric problem is mapped into a scalar minimization formulation. The mapping provides a straightforward method for converting abstract geometric concepts into a construct that is easily computed. The minimization approach is demonstrated in three application areas: computer animation, visualization, and physically-based modeling. In the computer animation application, cost minimization may be utilized to generate motion paths and joint parameters for animated actors. The approach may also be used to generate deformable models that extract closed 3-D geometric models from volume data for visualization. In the final application, the approach provides the fundamental structure to a physically-based model of woven cloth.

1 Introduction

A fundamental component of any three-dimensional (3-D) computer graphics application is a 3-D geometric model. The geometric model defines the object or objects that are being viewed and analyzed. By combining the geometric model with viewing, lighting and shading information, an image may be created. This is the process of rendering. In computer graphics, the geometric primitive of choice has been the polygon, with large models being defined as collection of polygons. More sophisticated applications may use more complex modeling primitives, such as quadrics, superquadrics, Bezier surfaces, and NURBS [11]. These primitives may be stitched together either with a boundary representation, or may be combined to produce a Constructive Solid Geometry (CSG) model [39]. One of the latest modeling primitives to be utilized by the computer graphics community is the volume data set [26]. In this representation, objects are defined as sets of 3-D points with a scalar value associated with each point. Algorithms have been developed to convert all of these sophisticated modeling primitives into polygons. This is necessary because most commercial computer graphics workstations contain special-purpose hardware for rapidly displaying polygons.

Frequently the creation of geometric models for a specific application can be time-consuming and difficult. Architectural models for walk-throughs, engine models for Computer-Aided Design (CAD) and volume data sets from medical imaging can be enormous and contain numerous important details. These details, at least in the case of CAD, are usually manually supplied by a designer. In addition to the effort required to produce a satisfactory geometric description, the model may not remain static or intact throughout the duration of the application. The model may be dynamic in the sense that the parameters defining its shape, position, orientation, or color may change over time. The changes may also be controlled and supplied by the designer or algorithmically produced by the application. The changes occurring in the geometric model most likely are not arbitrary, but very often are physically-based. In some applications a geometric model may by moved or transformed based on a set of physical laws.

When creating a complex 3-D computer graphics application, a program designer often faces numerous and difficult modeling tasks. A variety of tools may be utilized to solve the problems of generating, defining, and controlling animated models for advanced 3-D computer graphics applications. This paper presents a single technique, which may be employed to solve several classes of animated modeling problems. The technique involves mapping the 3-D geometric problem into a scalar minimization formulation. The efficacy of the technique is demonstrated for three problem domains: defining the motion of geometric objects, generating closed polygonal models from volume data, and defining the underlying structure of a physically-based model. The technique presented here is an application of and extension to a previously published algorithm, Energy Constraints on Parameterized Models by Witkin, Fleischer and Barr [60]. In the work described here, cost minimization (a more accurate term than energy minimization) has been used for more than just defining geometric constraints. It will be shown that cost minimization may be utilized to solve a number of 3-D animated modeling problems in computer graphics.

Cost minimization is just one of the tools that may be employed when addressing the 3-D animated modeling issues in a computer graphics applications. There are three main reasons for applying cost minimization to a specific 3-D geometric problem. First, mapping the problem into a scalar cost domain allows the application to easily compute the interactions of different primitives and data sets, potentially with different dimensionalities, in a straightforward unified way. Numerous 3-D geometric models, relationships and interactions may be converted into a cost function, providing a single representation for computing a simulation result. Dropping the dimensionality of complex geometric calculations effectively to one greatly simplifies the computations and allows for the development of general simulation tools which may then by applied to numerous and different scenarios. Second, for many problems in computer graphics a complex physically-based simulation that accurately describes geometry changing as a function of time is not required. For these problems, working in the energy/cost minimization domain, rather than the force integration domain, provides a simpler, straightforward approach to the problem. Finally, working in the optimization/minimization domain opens up the opportunity to apply the mature and well-studied techniques of this field to another application area. Optimization is a field that has been explored for many years. Thus robust and powerful algorithms are available for attacking the geometric problems of 3-D computer graphics.

The remainder of the paper first presents related work in optimization for computer graphics. The paper then describes how the concept of imposing geometric constraints by minimizing cost functions may be utilized and extended to accomplish a variety of animated modeling tasks for computer graphics. The minimization approach is demonstrated in three application areas: computer animation, visualization, and physically-based modeling. In the computer animation application, cost minimization may be utilized to generate motion paths and joint parameters for animated actors. The approach may also be used to generate deformable models (know as Geometrically Deformed Models, GDMs) that extract closed 3-D geometric models from volume data for visualization. In the final application, the approach provides the fundamental structure to a physically-based model of woven cloth. Results in each of these application areas are presented.

2 Previous Work

The techniques described here are motivated by and based on the work of Witkin et al. [60] who present a simple but general approach to imposing and solving geometric constraints on parameterized models. They define geometric constraints in terms of an "energy" function. The free variables of the function are the geometric parameters being constrained. The function is formulated such that the constraints are met when the ensemble of parameter values minimizes the function. Finding the parameter values which minimize the "energy" function imposes the geometric constraint. The minimum may be found by numerically calculating the gradient of the "energy" function and following it through the parameter space of the model to a minimum. The functions defined by Witkin et al. do not necessarily model the energy of actual physical systems.

Thus, using the term "cost" function is more accurate when referring to the functions used in this approach.

Minimization and optimization techniques have recently been employed extensively in computer graphics and related fields. Numerous researchers have utilized optimization techniques to assist in the construction and design of complex surfaces. Celniker and Gossard [9] developed a method for interactively designing fair shapes subject to user-defined constraints. Moreton and Sequin [38], and Halstead et al. [20] extended this approach to surfaces of arbitrary genus. Welch and Witkin [56, 57] described a technique for interactively defining an infinitely malleable surface which has mutable topology. Terzopolous and Qin [53] extend these techniques to NURB surfaces. Hoppe et al. [22] use mesh optimization [21] to automatically reconstruct accurate and concise piecewise smooth surface models from scattered range data. Witkin and Heckbert [62] employ particles and constrained optimization to sample and control implicit surfaces. Whitaker and Chen [58, 59] use an energy minimization process to transform implicitly represented deformable models which find iso-surfaces in 3-D sampled data. MacDonald et al. [33] have refined the GDM cost equations in order to identify and match anatomical structures in MRI data.

Terzopoulos et al. [25, 34, 50, 51, 52] have developed a class of 2-D and 3-D deformable models for use within computer graphics and computer vision. Their "deformable" models are a simplification of continuum mechanics models for flexible materials. From this foundation they derive energy functions for controlling the shape and behavior of the deformable models. In the 2-D case (Snakes) [25], the total energy over a flexible curve includes the internal deformation energy, potentials arising from interactions with images, and external potentials produced by user interactions. Minimizing the energy of the Snake drives it toward features of interest in the image. Snakes may be extended into 3-D by making several symmetry and viewing assumptions about objects in an image [50, 51]. This allows the Snake to expand into 3-D creating an axi-symmetric geometric model of an object in an image. 3-D models may also be generated from volume data using a dynamic "balloon" model, a spherical thin-plate under tension surface spline [34]. Here, a true 3-D closed deformable model is placed inside a volume data set in order to extract anatomical structures. The total energy in this case involves the deformation energy of a surface, interactions with the data set and user input, and an expansion term, which pushes out the "balloon" to find the appropriate object boundary. The same underlying techniques have also been used to produce animations of flexible materials that may bend, stretch, and even tear [52].

Optimization has not only been used in 3-D geometric modeling applications. Sederberg et al. [46, 47] developed two methods for blending one 2-D shape into another in a way that minimizes the "work" required for the transformation. Maillot et al. [32] proposed a technique that lowers the distortion produced during texture mapping. Kawai et al. [27] presented a method for designing the illumination of an environment using an optimization technique applied to a radiosity-based image generation system. Snyder [48] has shown that numerous problems in computer graphics may be formulated as constrained minimization problems, and therefore can be solved with interval analysis techniques. Platt and Barr [45] suggest that the simulation of physically-based flexible objects may be formulated as a constrained optimization problem. Several researchers [12, 42, 55] have utilized energy minimization to compute static solutions for draping simulations of cloth-like objects.

Optimization techniques have also been applied in the area of computer animation. Girard [16] makes use of optimization criteria to adaptively place feet and shift gaits during the locomotion of legged animals. Brotman and Netravali [8] provide a method for interpolating keyframes using optimal control, which minimizes the control energy needed to meet the constraints of the system. Barzel and Barr [2] describe "dynamic constraints", a technique for constrained assembly and control of computer graphics models using inverse dynamics. Spacetime constraints, initially developed by Witkin and Kass [61] and later extended by Cohen et al. [10, 31] and Ngo and Mark [41], is a technique for specifying the form and timing of physically-valid actions. Performing a constrained optimization produces motions that satisfy "what" constraints and optimize "how" criteria. Phillips, Zhao and Badler [43, 44] implemented kinematic constraints on articulated models with a minimization technique. Gleicher and Witkin [17] demonstrated a method for controlling a virtual camera by constraining features in the image seen through its lens. Snyder et al. [49] utilized an interval approach to detect collisions between time-dependent parametric and implicit surfaces. Van de Panne and Fiume [54] use stochastic minimization to find optimal values for the parameters of a sensor-actuator network used to animate virtual creatures. Grzeszczuk and Terzopoulous [19] created virtual creatures that learn realistic locomotion by optimizing a class of objective functionals. In robotics [40], Hwang and Ahuja [24], Khatib [28], and Krogh [30] used potential fields and minimization as a method for path planning.

3 Computer Graphics Cost Minimization Applications

This section demonstrates how the concept of imposing geometric constraints with cost functions may be applied and extended in order to solve animated modeling problems in three different computer graphics applications. Mapping these complex 3-D problems into a scalar minimization domain greatly simplifies the computation and allows for the application of well-developed optimization algorithms to 3-D modeling tasks. The three application areas are computer animation, visualization, and physically-based modeling. In the computer animation application, cost minimization is used to automatically generate goal-oriented motions. The visualization application demonstrates that a minimization approach may be used to produce deformable models that extract closed geometric models from image and volume data. Finally, in the physically-based modeling application, cost functions are used to describe the geometric relationships and constraints of a microstructural model of woven cloth.

The same general approach is taken in all three applications. First, the application is studied to determine which geometric constraints are required. The constraints usually consist of point-to-point constraints, that hold separate components together, angular constraints, formed at a common connection point, and a repulsion constraint for collision avoidance, which can be based on a geometric object or a set of sampled data. There may also be many other constraints based on a variety of geometric quantities, e.g., distance, length, area, and volume. Once the proper geometric constraints have been determined, a cost function is derived which enforces the constraint. The function is zero in the allowable range of the geometric quantity or at a specific quantity value that defines the constraint, e.g., a pre-determined length constraint. The function increases outside the allowable range or specific value. Each separate function is combined into a single large function, which is then minimized to produce the final result.

3.1 Choreographing Goal-Oriented Motion Using Cost Functions

3.1.1 Defining Key Goals

In the field of computer animation, imposing geometric constraints with cost functions may be applied to produce goal-oriented motion through the specification of “key goals” and “key situations” [4]. The cost function minimization process automatically moves the models of the animation from key goal to key goal, thus interpolating them to produce the complete animation. This process differs from traditional key-framing in that the animator specifies a higher-level goal, such as “reach for this object there” and “move from here to there”. The animator does not need to specify the modeling parameters at the key goals. These are automatically calculated by the minimization process. Key goals may also contain inherent constraints such as “do not bend your elbow past a certain angle”, “do not collide with another object”, or ”do not move out this room”. In this case the constraints are never strictly enforced, but act more like penalties. The “constraints” here simply constrain the motion of the moving objects by penalizing certain actions or movements. Working in the cost domain easily supports the combination of goals and penalties into a single formulation. Each goal and each penalty is simply a term in the cost function. A weight associated with each term defines its relative importance in the overall key goal.

In this usage, cost functions are not only used to define geometric constraints, but are used to encapsulate high-order activities. They can be used to define goal-oriented motions and actions. This alternate application of cost functions centers around the motion that occurs during the minimization process. A cost function is defined whose variables are the animated parameters of a scene. The parameters are modified in such a way as to minimize the cost function. The minimum cost configuration can be viewed as a ”key goal” configuration. The values of the parameters are stored at certain intervals during the cost function minimization. This produces a path through the parameter space of the model being animated. By incrementally moving along the parameter space curve and updating the model defined by the parameters, an animation of the model performing a goal-oriented action may be produced.

Viewing cost functions as tools for defining goal-oriented motions makes the process of finding the minimum more important than actually maintaining the minimum. It is the changing of the animated parameters during the minimum-finding process which produces the goal-oriented action. Defining actions in this fashion simplifies the action specification task. It becomes simply a matter of defining a function. The disadvantage is that the animator surrenders a significant amount of control over the details of the motion. In return the animator is provided with a powerful high-level tool for specifying complicated actions. The animator specifies what must happen, not how it happens. The technique determines the details of the "how", when finding the minimum of the function. With cost functions the idea of creating goal-oriented complex motions is easily implemented, freeing the animator from significant amounts of tedious work with intricate modeling details.

3.1.2 Two Animation Examples

A first, straightforward example of cost minimization for computer animation involves planning the path for an animated object. This involves moving through an environment while avoiding the objects in that environment. The initial conditions for this scenario include the start and end positions of the animated object and the positions (and possible motions) of the other objects in the scene. A cost function that encapsulates the collision-free movement of an object through a cluttered environment is

$$C_{PathPlan}(x,y,z) = \mid P(x,y,z) - P_{goal} \mid + \sum_{i=1}^{n} field(dist(P(x,y,z), obj_i)), \qquad (3.1)$$

where $P(x,y,z)$ is the position of the animated object, P_{goal} is the location to which it is moving, and $dist(P(x,y,z), obj_i)$ is a function which calculates the distance between the current position of the animated object $(P(x,y,z))$ and another object (obj_i) in the environment. The $field$ function provides a penalty field for collision avoidance, and is calculated for the n obstacles in the environment. It is zero at a specified distance

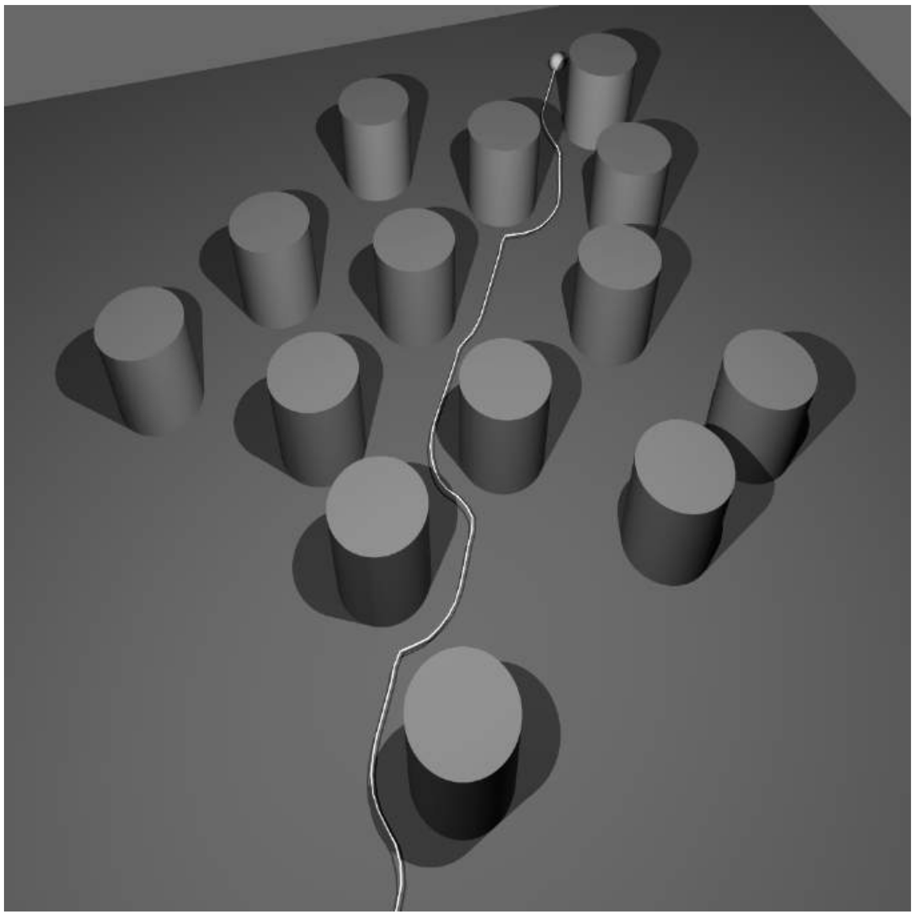


Figure 1: A collision-free path.

R, and goes to infinity when the distance goes to zero. One such function is

$$field(d) = \begin{cases} ln(R/d) & 0 < d \leq R \\ 0 & d > R. \end{cases} \tag{3.2}$$

The cost function may be minimized using a gradient descent algorithm [15]. The gradient of equation 3.1, $\nabla C_{PathPlan}$, can be approximated by $\Delta C_{PathPlan}/\Delta P$. The negative gradient points in the direction of decreasing cost. Moving the animated object in this direction minimizes the cost function and brings it closer to achieving the desired goal. As seen by the path automatically generated in Figure 1, minimizing

Equation 3.1 moves an object toward a goal position until it comes in contact with the penalty field of an obstacle. The penalty field alters the direction of the object, so that it will move around the obstacle until it may proceed directly to the goal. The result was produced with a general cost minimization capability that was implemented inside an object-oriented computer animation system [3, 14].

Another, more complex computer animation example involves the motion of a constrained articulated structure. In this case a robot arm is used, but the function used here is applicable to any model that contains a hierarchical jointed structure. Here, an animated object (actor) may be directed to reach for an object. Naturally, the actor should not collide with any other object in the environment, and the rotations of the arm are constrained. The arm's joints cannot bend to any arbitrary orientation. A cost function that may be used for directing a constrained articulated object to reach for a goal point is

$$\begin{aligned} C_{reach}(\phi_0, \phi_1, \phi_2, \phi_3) &= \mid P_{tip}(\phi_0, \phi_1, \phi_2, \phi_3) - P_{goal} \mid \\ &\quad + \sum_{i=1}^{n} field(dist(P_{tip}(\phi_0, \phi_1, \phi_2, \phi_3), obj_i)) \\ &\quad + \sum_{j=0}^{3} limit(\phi_j). \end{aligned} \tag{3.3}$$

Equation 3.3 has several components, some of which it has in common with Equation 3.1. The first component $\mid P_{tip} - P_{goal} \mid$ drives the tip of the arm towards the goal position. P_{tip} is a complex function of the arm's joint parameters, $\phi_0, \phi_1, \phi_2, and\ \phi_3$. Since the gradient is numerically approximated during cost minimization, this detail is unimportant and may be hidden inside a function call or method execution. The second

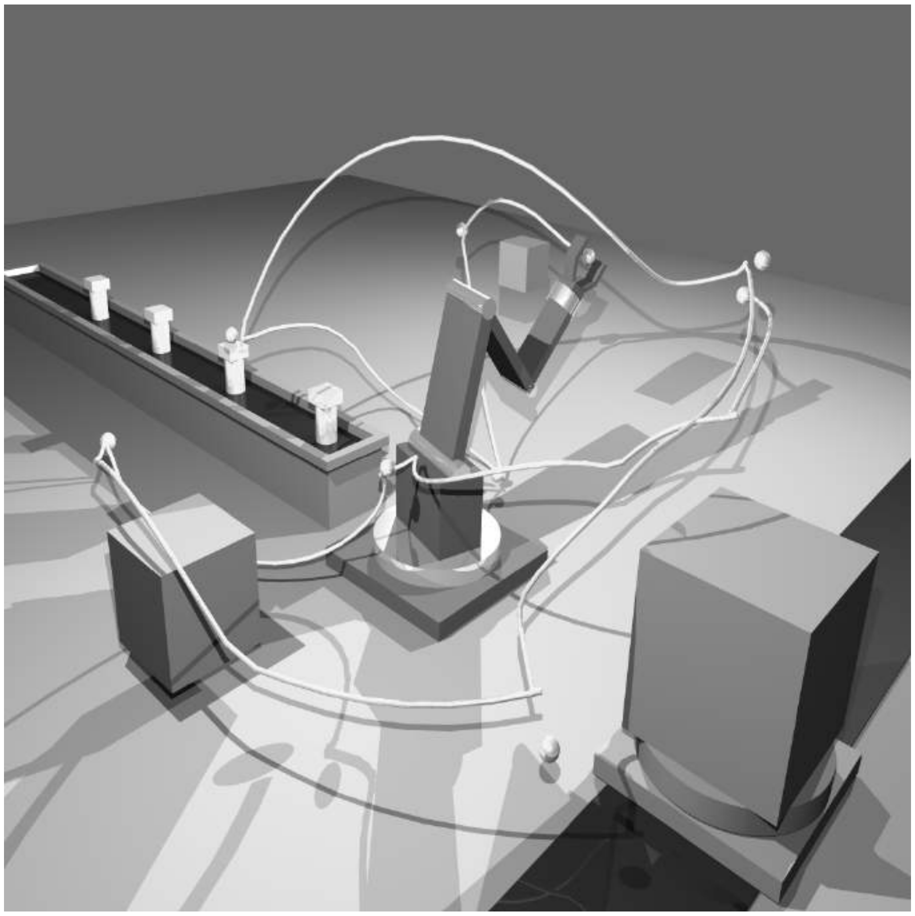


Figure 2: A collision-free and constrained motion for an articulated object

component pushes the tip of the arm away from the obstacles in the environment with the same *field* and *dist* function used in Equation 3.1. The final component implements joint value constraints by summing the *limit* function for each of the joints. The *limit* function increases as joint angle ϕ_j approaches either its maximum or minimum limit and is zero otherwise. This strongly inhibits each joint from bending beyond its prescribed limits. One such function which enforces joint limits is

$$limit(\phi) = \begin{cases} ln(\delta/(\phi - MIN)) & MIN < \phi \leq MIN + \delta \\ 0 & MIN + \delta < \phi < MAX - \delta \\ ln(\delta/(MAX - \phi)) & MAX - \delta \leq \phi < MAX, \end{cases} \tag{3.4}$$

where ϕ is the angle of the current joint, MIN and MAX are the minimum and maximum allowable angles for a particular joint, and δ is the angular distance from the limits at which the limit function becomes non-zero.

Once again a gradient descent algorithm is used to modify the joint angles of the arm in order to drive Equation 3.3 to a minimum. The result of the minimization is presented in Figure 2 and in a computer-generated animation [5]. In this example the arm is directed to move to a series of goals points, represented by spheres, in a cluttered environment. As it reaches one goal point, it immediately moves to the next one, while moving around the obstacles in the scene. The tube hanging in space represents the path traced out by the tip of the arm as it moves. As can be seen by the fact that the tube does not go through the sphere in the lower center of Figure 2, not all goal points can be reached. In this case, the goal is unattainable due to the robot's joint limits, and the arm reaches to the closest possible point before moving on to the next goal.

3.2 Geometrically Deformed Models

In visualization it often necessary to display 2-D image and 3-D volume data sets. There are a variety of techniques for accomplishing this task. One common method involves first converting the sampled data set into a geometric model. The geometric model is then easily displayed. The process of generating a model from a sampled data set normally includes a segmentation stage. One technique for generating a geometric model from a sampled volume data set utilizes Geometrically Deformed Models (GDMs) [35, 36, 37].

In medicine, it is recognized that significant quantitative information is imbedded in the volume data sets produced by medical equipment such as MRI, Ultra-Sound, CAT and PET scanners. Experience has shown that doctors want more than just pictures from these data sets. They wish to measure the objects being visualized. When a tumor or other abnormality is encountered, it is useful to know if its size changes over the duration of a particular treatment. GDMs were developed to meet these measurement needs, as well as the normal requirements of visualization. GDMs have several features which make them especially attractive for generating models from sampled volume data for both visualization and geometric measurements. GDMs produce topologically-closed geometric models that may be used to calculate the volume and surface area of topologically-spherical objects. GDMs can easily cope with noisy and incomplete data, filling in gaps, and working around spikes and troughs of noise. They also produce fewer polygons than other methods, thus providing models that require less time to render.

A GDM is a topologically-spherical and -closed model represented as a connected set of points. The process of utilizing a GDM to extract a model from sampled data begins by placing the GDM either within or around the structure to be extracted in the data set. The GDM is deformed, while maintaining a set of constraints, to fit to the object in the data set. The deformation is achieved by defining a function at each vertex of the GDM that associates costs with local movements, interactions with the data set, and local geometric deformations. Minimizing the cost function defined for each GDM vertex causes the GDM to expand or contract into the shape of the object imbedded in the data set.

Three cases where GDMs have been utilized to extract closed geometric models from sampled data sets are presented. The cost functions defined at each vertex of the GDM are described for the three cases. The first case involves finding the boundary of a lung cavity from a single MRI scan of a person's chest. In the second and third cases 3-D closed geometric models of a nerve cavity of a tooth and a cooling chamber in a turbine blade are extracted from volume data sets. In all three cases an adaptive gradient descent algorithm [15] is used to minimize the cost functions. Other minimization techniques that would prevent the solutions from being caught in local minima, e.g., simulated annealing [29], or genetic algorithms [18], were considered. Since the path taken during the minimization is also important, especially in the computer animation application described in the previous section, as well as the final solution, these other techniques were not deemed necessary, because they might produce discontinuous sets of parameter values. In any event, being trapped in an unsatisfactory local minima has not been a problem in the course of this work.

3.2.1 Finding a Lung Cavity

In the first case a 2-D GDM is used to find an ill-defined boundary in an image data set [35]. In Figure 3 a ring of 50 points is placed inside a single MRI scan of a human's chest. A cost function C is associated with each vertex of the ring. C is defined as

$$C = a_0\mathcal{I} + a_1\mathcal{D} + a_2\Lambda + a_3\Theta, \tag{3.5}$$

where $\mathcal{I}$ embodies the point's interaction with the image data, $\mathcal{D}$ is the potential field that drives the GDM towards the object boundary, Λ attempts to maintain a distance constraint between neighboring points, Θ maintains angular limits between neighboring points, and the a_i's are scaling factors.

$\mathcal{D}$ is a potential that expands the GDM, pushing it toward the boundary of the lung cavity. $\mathcal{I}$ is a potential based on the image data which pushes back against $\mathcal{D}$. As the image intensity increases, the force pushing back against the GDM increases. Λ and Θ are the cost functions that maintain the geometric and topological integrity of the model, preventing it from locally bending in on itself, and from abnormally stretching at any one place. Conceptually, the image data can also be thought of as a three-dimensional terrain where the intensities are elevations. The GDM is a flexible ring that is expanding inside the terrain, and finds some equilibrium between the force pushing it out, the "gravitational" potential associated with moving up into the terrain, and the internal potentials of deformation.

For the case presented in Figure 3, $\mathcal{I}$ is defined as

$$\mathcal{I}(x,y) = \begin{cases} 0 & I(x,y) < T \\ I(x,y) - T & I(x,y) \geq T, \end{cases} \tag{3.6}$$

where $I(x,y)$ is the bilinearly interpolated intensity value of the image at position (x,y) within the image, and T is a user-defined threshold value. $\mathcal{D}$ is defined as

$$\mathcal{D}(x,y) = ln \frac{M}{|(x,y) - (x_0,y_0)|}, \tag{3.7}$$

where (x_0, y_0) is the initial center of the ring, and M is the maximum distance to the

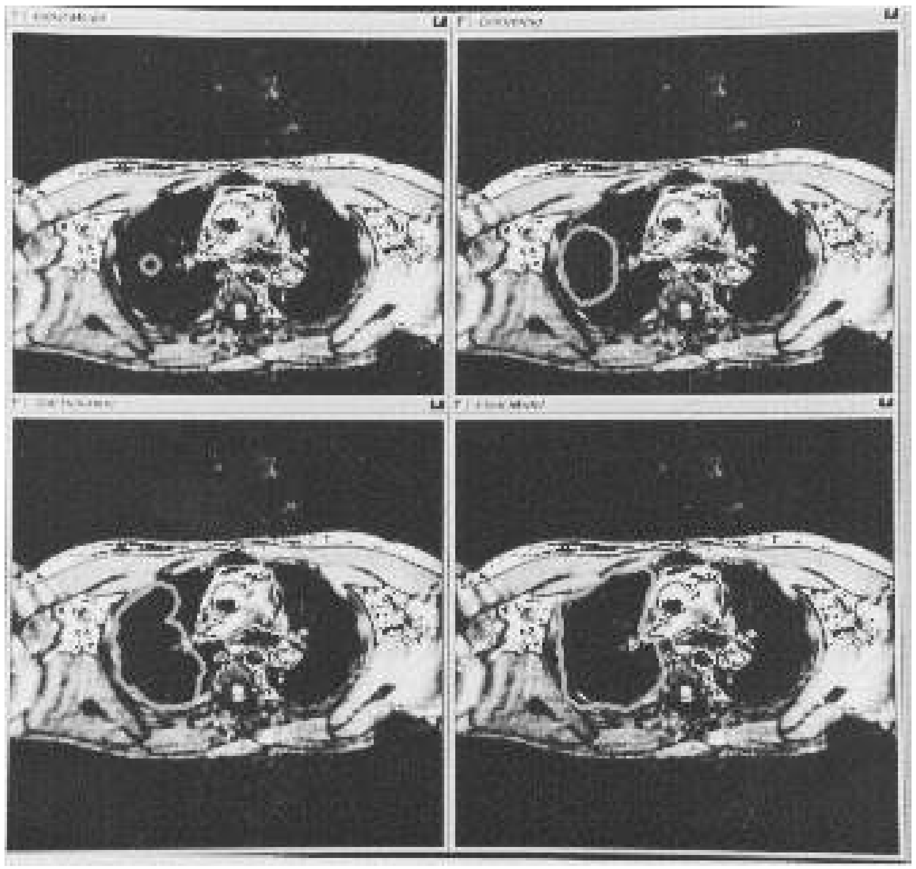


Figure 3: Four stages of finding a lung cavity with a GDM

edge of the image. Λ is defined as

$$\Lambda(d) = \begin{cases} .8\mu - d & d < .8\mu \\ 0 & .8\mu \leq d \leq 1.2\mu \\ d - 1.2\mu & d > 1.2\mu, \end{cases} \tag{3.8}$$

where d is the distance between the current point and a neighboring point, and μ is the mean distance between the current point's four nearest neighbors. The function allows the inter-vertex distance d to freely range within the value of $d \pm .2\mu$. Once a vertex begins to move outside of this range, a linear cost restrains it. Θ is defined as

$$\Theta(\alpha) = \begin{cases} .85\alpha_0 - \alpha & \alpha < .85\alpha_0 \\ 0 & .85\alpha_0 \leq \alpha \leq 1.15\alpha_0 \\ \alpha - 1.15\alpha_0 & \alpha > 1.15\alpha_0, \end{cases} \tag{3.9}$$

where α is the angle formed between a GDM vertex and its two immediate neighbors. α_0 is the angle formed at each GDM vertex at initialization time, and is a function of the number of points used to define the ring. The angle α is constrained in a similar fashion to the inter-vertex distance. A vertex's angle may freely vary within an $\alpha \pm .15\alpha_0$ range. Bending outside this range incurs a linear cost.

Figure 3 presents four stages in the minimization process. In the first stage, the ring of 50 points is placed inside the image. In the second stage, the ring has begun to expand and it has been caught on a spike of noise. In the third stage, the GDM has worked its way around the noise, and is proceeding toward the lung boundary. In the final stage, the GDM has come to an equilibrium that approximates the boundary of the lung cavity. Even though the edge of the cavity is ill-defined and contains holes, the geometric constraints of the approach prevent the GDM from leaking or stretching out from the cavity.

3.2.2 Extracting 3-D Models from Volume Data Sets

When going to 3-D the basics of GDMs do not change. In 3-D, GDMs are a collection of points organized into a topologically-spherical and -closed object, that deforms as it expands or contracts in response to the 3-D sampled data in which it is imbedded [36, 37]. In 3-D, a GDM, initially represented by 12 points formed as an icosahedron, is placed inside the volume data set. The cost function

$$C = a_0 D + a_1 I + a_2 T \tag{3.10}$$

is defined for each point, where D is a more general expansion deformation, I is the thresholded tri-linearly interpolated value of the data set at (x, y, z), T is an approximate measure of the local curvature, which also maintains the local topological integrity of the GDM, and the a_i's are scaling factors. Minimizing C for each point, once again, deforms the GDM, fitting it to the object imbedded in the data set.

For the examples presented in Figure 5 and 6, D is defined at each vertex as a potential that moves the vertex in the direction of the local surface normal to the GDM. The normal N may be approximated by averaging the normals of all the faces shared by the current vertex. The expansion is achieved by defining a point P_0 in the direction of the normal N at a distance t from the current vertex P, and associating an attracting potential to it. P_0 is defined as

$$P_0 = P + Nt. \tag{3.11}$$

D may then be defined as

$$D = |(P_0 - P)|. \tag{3.12}$$

Constraining a GDM's local curvature prevents each vertex from straying too far from its neighbors. The relationship is graphically presented in Figure 4. The ratio of the distance from the current vertex to the centroid of its neighbors and the maximum

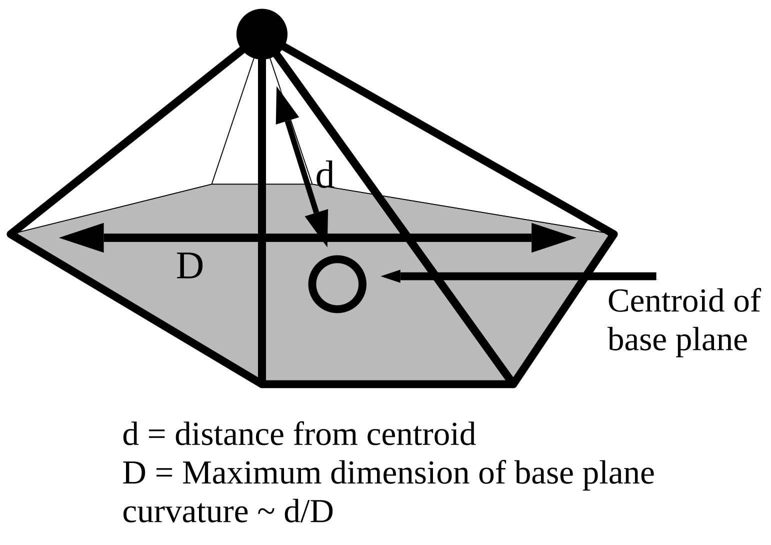

d
D
Centroid of base plane
d = distance from centroid
D = Maximum dimension of base plane
curvature ~ d/D

Figure 4: Curvature approximation in GDMs

distance between its neighbors gives a rough measure of the local curvature. This ratio defines the cost function term T,

$$T = \frac{|(x,y,z) - \frac{1}{n}\sum_{l=1}^{n}(x_l,y_l,z_l)|}{\max(|(x_j,y_j,z_j) - (x_k,y_k,z_k)|)} \tag{3.13}$$

where (x,y,z) is the position of the current vertex, n is the number of neighbors, and (x_j,y_j,z_j), (x_k,y_k,z_k) are the neighbors of the current vertex, $1 \le j < n, j < k \le n$. This function directs the vertex towards the centroid of its neighbors, which attempts to make the model locally flat.

Figures 5 and 6 present the results of the minimization process on two different data sets. In Figure 5, a model of a nerve cavity of a tooth is extracted from a CT scan. In Figure 6, a GDM expands to create a closed geometric model of a cooling chamber of a turbine blade. Only the GDM is shown, without attempting to display the volume data directly. It should also be noted that the GDM is initially defined as an icosahedron. New points and faces are added during the expansion in areas where the faces exceed a

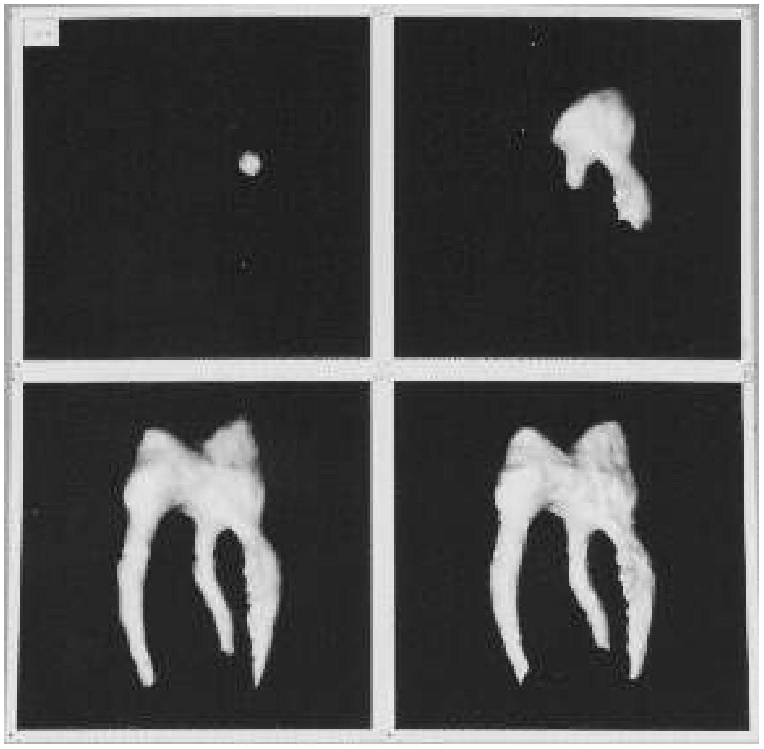


Figure 5: Four stages of extracting a nerve cavity with a GDM

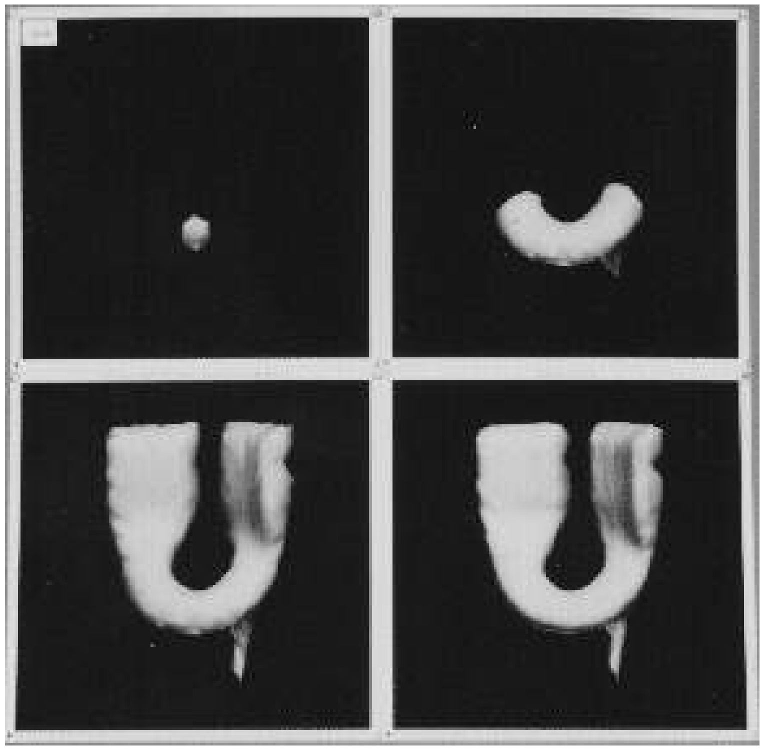


Figure 6: Four stages of extracting a cooling chamber with a GDM

predefined area. This subdivision process, as well as the specific details of the equations and their parameters, are fully explained in [36].

The results in this section are clearly similar to the "Snakes" and dynamic "balloon" work of Terzopoulos et al. [25, 34]. The GDM method provides a simpler, more geometrically-intuitive approach to the problem of extracting closed shapes from 2-D image and 3-D volume data. More importantly, 2-D GDMs do not need to be placed near a feature or assisted by a user to move around noise. 2-D GDMs assume that some approximately closed structure exists in the image, and it searches it out. The user only needs to initially place the GDM somewhere inside the structure. The GDM then automatically finds the object boundary in the presence of noise and holes. In 3-D, the GDM approach provides the same functionality as the dynamic "balloon" approach. Its simpler formulation easily supports such features as adaptive subdivision, a feature difficult to implement in the more complex finite-element-based "balloon" method.

3.3 A Particle-Based Model of Woven Cloth

The final example demonstrates that cost functions may be utilized to define the microstructure of a physically-based model of a complex material, woven cloth [6, 7]. The mechanical behavior of the material may also be formulated as an energy function. In this case, the term "energy" may be used, since the "energy" of the phenomenon is being modeled. Having both the material's microstructure and mechanical properties defined with cost/energy functions makes it possible to compute simulation results with a single energy minimization scheme. Since all aspects of the model are mapped

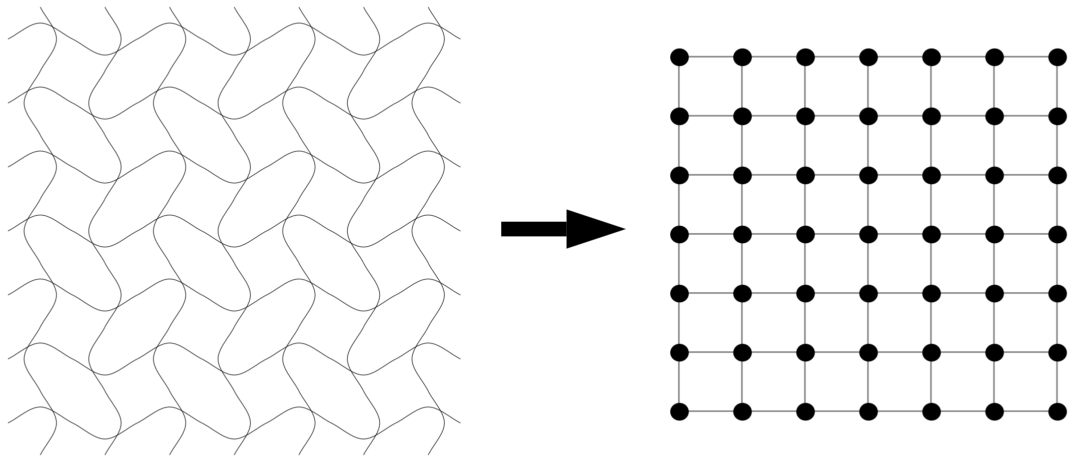


Figure 7: Mapping a plain weave to a particle grid

into an energy domain, the components which describe the underlying topology of the model and its mechanical behavior may be handled in a consistent manner.

Woven cloth may be modeled as a collection of particles that conceptually represents the crossing points of warp and weft thread in a plain weave, as diagramed in Figure 7. Important mechanical interactions that determine the behavior of woven fabric occur at these points. Most significantly, the tension is typically so great at crossings that the threads are clamped together, providing an axis around which bending can occur in the plane of the cloth. Other more distributed interactions, such as stretching of threads and out of plane bending, can be conveniently discretized and lumped at the crossing points.

3.3.1 Model Components

In the model, the various thread-level structural constraints are represented with energy functions that capture simple geometric relationships between particles within a local neighborhood. These energy functions are meant to encapsulate four basic low-level interactions: thread collision, thread stretching, out-of-plane bending, and trellising. These are shown graphically in Figure 8, and are captured in the energy equation for particle i,

$$U_i = U_{repel_i} + U_{stretch_i} + U_{bend_i} + U_{trellis_i} + U_{grav_i}. \tag{3.14}$$

In this equation, U_{repel_i} is an artificial energy of repulsion, that effectively keeps every other particle at a minimum distance, providing some measure of thread collision detection and helping prevent self-intersection of the cloth. $U_{stretch_i}$ captures the energy of tensile strain between each particle and its four-connected neighbors. U_{bend_i} is the energy due to threads bending out of the local plane of the cloth, and $U_{trellis_i}$ is the energy due to bending around a thread crossing in the plane. U_{grav_i} is the potential energy due to gravity. Repelling and stretching are functions only of interparticle distance r_{ij} (Figure 8-Ia), whereas bending and trellising are functions of various angular relationships between the segments joining particles (Figure 8-IIa and 8-IIIa). U_{grav_i} is a function of the height of the particle. Trellising occurs when threads are held fast at a crossing and bend to create an "S-curve" in the local plane of the cloth, and is related to shearing in a continuous sheet of material, but since the model treats cloth as an interwoven grid of threads, trellising is a more descriptive term.

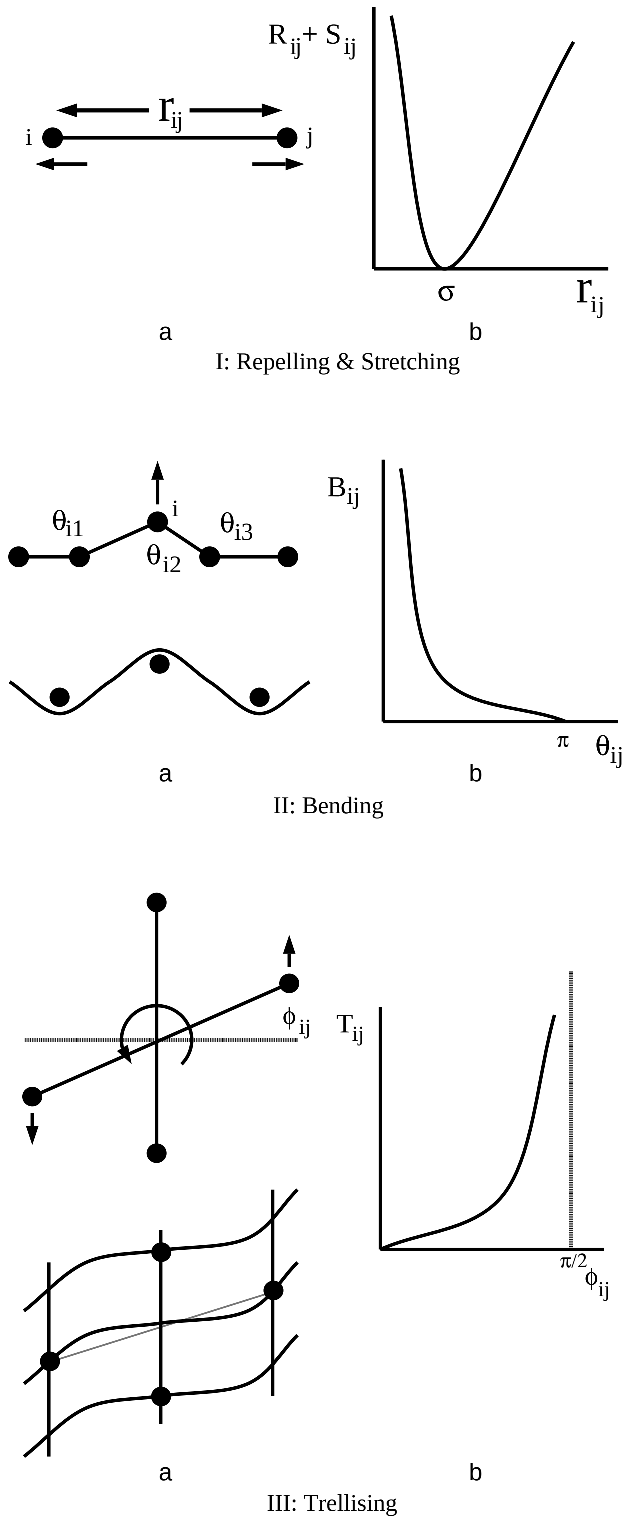


Figure 8: Cloth model energy functions

It is assumed that the threads in a woven fabric do not stretch significantly when a cloth is simply draping under its own weight. Therefore, the combined stretching and repelling energy function $R+S$ shown in Figure 8-Ib is not empirical, and is meant only to provide collision prevention and a steep energy well that acts to tightly constrain each particle to a nominal distance σ from each of its 4-connected neighbors. These functions literally hold the particle grid together and inhibit self-intersection. They are defined as

$$R(r_{ij}) = \begin{cases} C_0[(\sigma - r_{ij})^5/r_{ij}] & r_{ij} \leq \sigma \\ 0 & r_{ij} > \sigma, \end{cases} \tag{3.15}$$

and

$$S(r_{ij}) = \begin{cases} 0 & r_{ij} \leq \sigma \\ C_0[((r_{ij} - \sigma)/\sigma)^5] & r_{ij} > \sigma, \end{cases} \tag{3.16}$$

where C_0 is a scale parameter.

The function U_{repel_i} prevents collision and self-intersection, so it is calculated by summing over all particles, as given by

$$U_{repel_i} = \sum_{j \neq i} R(r_{ij}). \tag{3.17}$$

In practice, the simulation algorithm maintains a spatial enumeration, so that the summation need only be done over spatial near-neighbors. An energy well is produced by directly coupling each particle with the stretching function S only to its 4-connected neighbors, as given by

$$U_{stretch_i} = \sum_{j \in N_i} S(r_{ij}), \tag{3.18}$$

where N_i is the set of particle i's four-connected neighbors.

The particle energy due to gravity is simply defined as

$$U_{grav_i} = m_i g h_i, \tag{3.19}$$

where h_i is the height of particle i, g is gravitational acceleration, and m_i is the mass of the small patch of cloth represented by the particle.

In contrast to stretching, it is assumed that bending and trellising are the significant contributors to the overall drape of cloth, when it is simply draping under its own weight. A graph of the bending energy B is shown in Figure 8-IIb which is a function of the angle formed by three particles along a weft or warp "thread line", as shown in Figure 8-IIa. The complete bending energy is

$$U_{bend_i} = \sum_{j \in M_i} B(\theta_{ij}), \tag{3.20}$$

where M_i is the set of six angles θ_{ij} formed by the segments connecting particle i and its eight nearest horizontal and vertical neighbors. This definition is used so that the total change in bending energy due to the change in the position of particle i is calculated in one formulation. It is the total change in energy which is needed to calculate the derivative ultimately utilized during the minimization process.

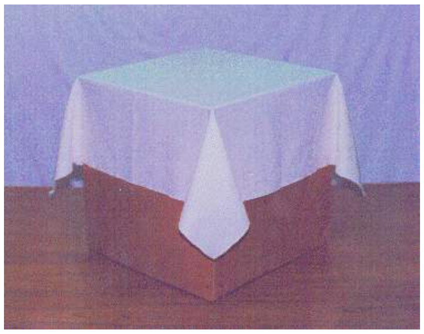
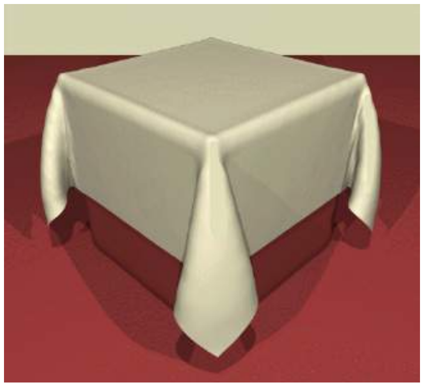
The phenomenon of trellising is diagramed in Figure 8-IIIa and a corresponding graph of the trellising energy T is shown in Figure 8-IIIb. Two segments are formed by connecting the two pairs of neighboring particles surrounding a central particle. An equilibrium crossing angle of 90° is assumed, but this angle could easily change over the course of a simulation due to frictional slippage. The trellis angle ϕ is then defined as the angle formed as one of the line segments moves away from this equilibrium. The complete function for the energy of trellising is

$$U_{trellis_i} = \sum_{j \in K_i} T(\phi_{ij}), \tag{3.21}$$

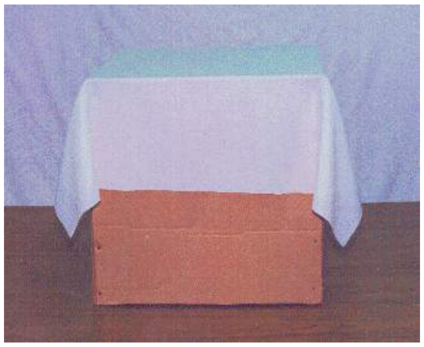
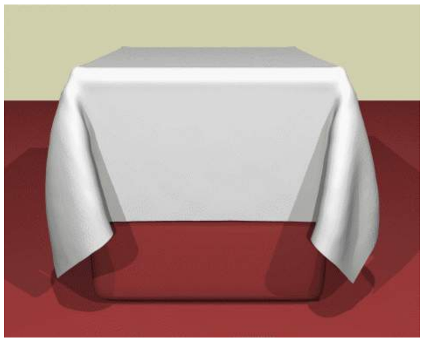
where K_i is the set of four trellising angles ϕ_{ij} formed around the four-connected neighbors of particle i. As with bending, this redundant formulation was chosen so that change in total energy with change in the particle's position is completely accounted for locally. Both $B(\theta_{ij})$ and $T(\phi_{ij})$ may be derived from empirical data, and describe the mechanical behavior of the material. Since they do not enforce geometric constraints, they are not detailed here. The details of their derivation may be found in earlier publications [6, 7].

3.3.2 Simulation Results

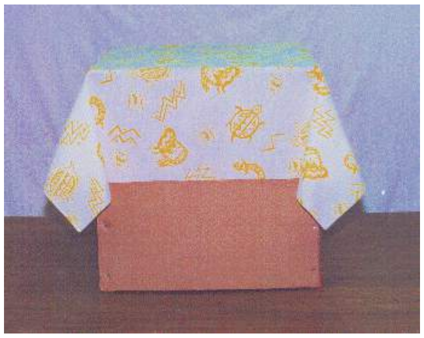
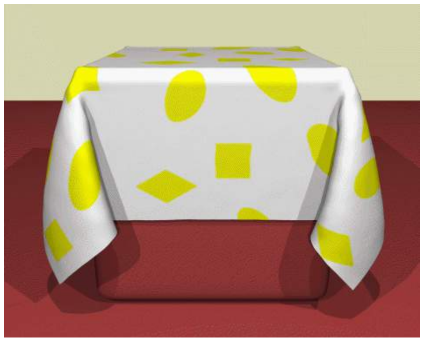
Figure 9 presents the results of three simulations, using bending and trellising energy functions derived from test data describing the mechanical properties of cotton, wool, and cotton/polyester cloth. The left column of Figure 9 contains photographs of the tested materials draped over a cube. The right column contains the associated simulated cloth drapings. The specific energy functions used in these simulations are detailed in [6]. The cloth sample is initially flat and placed above a cube. A mixed dynamic/minimization technique is used to place the cloth in contact with the cube in a partially-draped form [23]. A pure energy minimization using stochastic gradient descent [1, 13] drives the model into a final equilibrium configuration. Stochastic gradient descent is used here to prevent the solution from falling into an unwanted local actual virtual

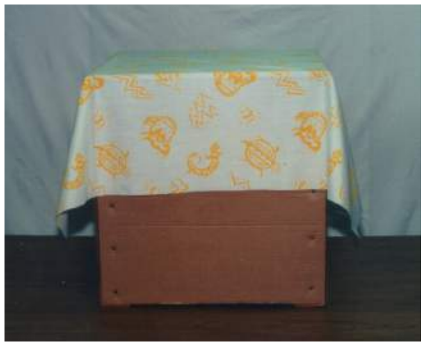
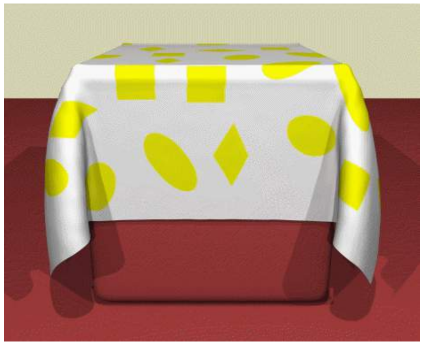



100% Cotton

100% Wool

Polyester/Cotton Blend

Figure 9: Real and simulated draping cloth

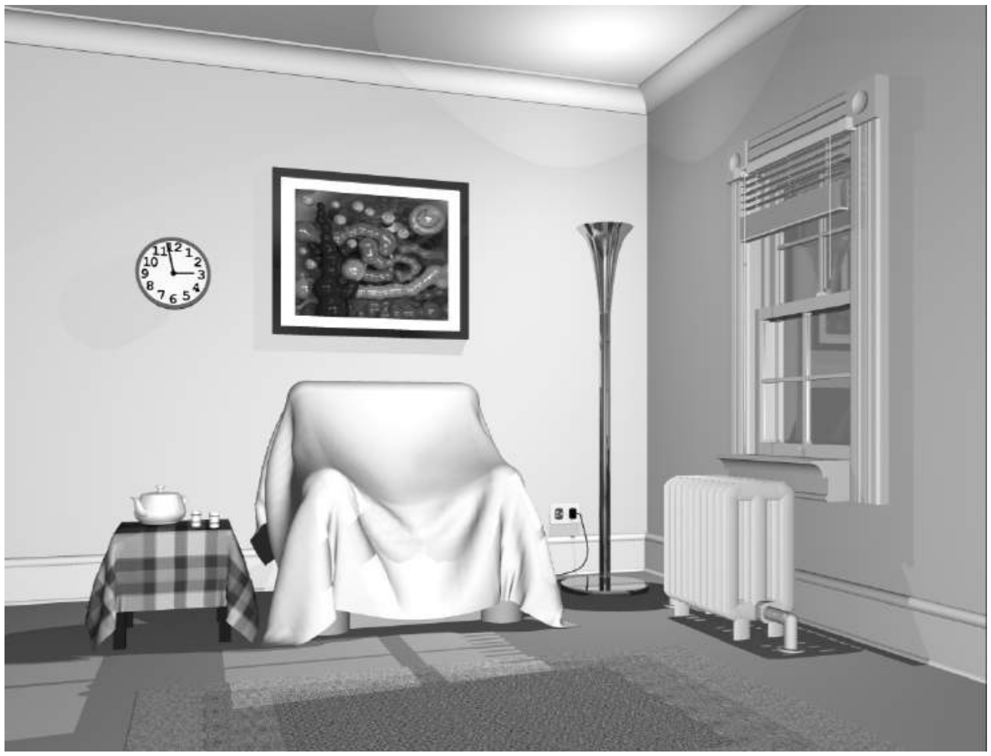


Figure 10: Cloth simulations in an interior setting

.

minimum, and to incorporate natural perturbations throughout the model. Additional cloth simulations within an interior setting are also presented in Figure 10.

The R and S functions hold the model together and provide its basic structure, while the B and T functions make the essential contributions to the draping behavior. It can be seen that the simulation results reasonably match the drapings of the real cloth on which the cloth-specific models are based. Characteristic macroscopic draping structures are reproduced by the simulations. In the cotton sample a single symmetric structure forms at a 45° angle from the corner. The wool sample is less stiff and the draping structure collapses into two lobes. The polyester/cotton blend has asymmetric bending properties and produces a draping structure that folds around the corner of the cube. With this new model, real materials can be measured, and the measured data may now be used to derive energy equations, allowing the draping behavior of specific materials to be confidently simulated on a computer.

4 Conclusion

This paper describes how the concept of modeling geometric constraints with cost functions may be utilized for a variety of animated modeling tasks in computer graphics. Mapping 3-D geometric modeling problems into a scalar minimization problem is a powerful technique that provides a consistent framework in which numerous 3-D geometric attributes, entities, and interactions may be brought together. This concept is demonstrated for three application areas.

The first example presents a method for creating goal-oriented motions for computer animation. Cost minimization provides path planning and inverse kinematics, combined with parameter limits and obstacle avoidance. In the second application, a technique for extracting closed geometric models from sampled data sets for visualization and quantitative analysis is presented. Cost functions are used to deform a geometric model, maintain the topological and geometric integrity of the model, and to define its interaction with a discrete data set. The final example presents a physically-based model of woven cloth, where cost functions embody and maintain the topology of the model, and prevent it from intersecting with itself. Energy functions are then used to define its mechanical properties. Since constraints and mechanical properties are defined in a cost/energy domain, they both can be computed in a unified and consistent manner. Clearly, defining 3-D geometric constraints with cost functions is a flexible and general technique that may be used to solve a number of animated modeling problems in computer graphics.

5 Acknowledgements

The work described in this paper was conducted over several years, in several laboratories with numerous collaborators, while I was employed at the Rensselaer Design Research Center. The computer animation research was conducted while the author was a visiting research engineer at the Fraunhofer Institute in Darmstadt, Germany. The visualization research was conducted with James V. Miller, William Lorensen, Robert O'Bara, and Michael Wozny, in conjunction with the General Electric Corporate Research and Development Center. James Miller significantly contributed to the work described in Section 3.2. The cloth modeling research was conducted with Donald House, from Williams College, Phillip Getto and Michael Wozny. Figure 10 was produced by Gene Greger and David Breen. This work was financially supported by the Industrial Associates Program of the Rensselaer Design Research Center, DLA contract No. DLA900-87-D-0016, and NSF grant No. CDR-8818826.

References

[1] F. Aluffi-Pentini, V. Parisi and F. Zirilli, "Global Optimization and Stochastic Differential Equations," *Journal of Optimization Theory and Applications*, Vol. 47, No. 1, pp. 1-16, September 1985.

[2] R. Barzel and A.H. Barr, "A Modeling System Based on Dynamic Constraints" *Computer Graphics* (Proc. SIGGRAPH), Vol. 22, No. 4, pp. 179-188, August 1988.

[3] D.E. Breen, "The Cost Analysis Object in The Clockworks," RDRC Technical Report TR-88048 (Center for Advanced Technology, Rensselaer Polytechnic Institute, Troy, NY, November 1988).

[4] D.E. Breen, "Choreographing Goal-Oriented Motion Using Cost Functions," *State-of-the-art in Computer Animation* (Proc. Computer Animation '89), eds. N. Magnenat-Thalmann and D. Thalmann (Springer-Verlag, Tokyo, June 1989) pp. 141-151.

[5] D.E. Breen and P.H. Getto, "Choreographing Goal-Oriented Motion Using Cost Functions," RDRC Videotape VT-9001 (Center for Advanced Technology, Rensselaer Polytechnic Institute, Troy, NY, 1990).

[6] D.E. Breen, "A Particle-Based Model for Simulating the Draping Behavior of Woven Cloth," Ph.D. Thesis, RDRC Technical Report TR-93011 (Center for Advanced Technology, Rensselaer Polytechnic Institute, Troy, NY, June 1993).

[7] D.E. Breen, D.H. House and M.J. Wozny, "Predicting the Drape of Woven Cloth Using Interacting Particles," *Computer Graphics* (Proc. SIGGRAPH), pp. 365-372, July 1994.

[8] L.S. Brotman and A.N. Netravali, "Motion Interpolation by Optimal Control," *Computer Graphics* (Proc. SIGGRAPH), Vol. 22, No. 4, pp. 309-315, August 1988.

[9] G. Celniker and D. Gossard, "Deformable Curve and Surface Finite-Elements for Free-Form Shape Design," *Computer Graphics* (Proc. SIGGRAPH), Vol. 25, No. 4, pp. 257-266, July 1991.

[10] M.F. Cohen, "Interactive Spacetime Control for Animation," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 293-302, July 1992.

[11] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design* (Academic Press, Boston, 1990).

[12] C.R. Feynman, *Modeling the Appearance of Cloth*, M.S. Thesis, Massachusetts Institute of Technology, 1986.

[13] S. Geman and C.-R. Hwang, "Diffusions for Global Optimization," *SIAM Journal of Control and Optimization*, Vol. 24, No. 5, pp. 1031-1043, September 1986.

[14] P.H. Getto and D.E. Breen, "An Object-Oriented Architecture for a Computer Animation System," *The Visual Computer*, Vol. 6, No. 2, pp. 79-92, 1990.

[15] P. Gill, W. Murray and M. Wright, *Practical Optimization* (Academic Press, London, 1981).

[16] M. Girard and A.A. Maciejewski, "Computational Modeling for the Computer Animation of Legged Figures," *Computer Graphics* (Proc. SIGGRAPH), Vol. 19, No. 3, pp. 263-270, July 1985.

[17] M. Gleicher and A. Witkin, "Through-the-Lens Camera Control," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 331-340, July 1992.

[18] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[19] R. Grzeszczuk. and D. Terzopoulos, "Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction," *Computer Graphics* (Proc. SIGGRAPH), pp. 63-70, August 1995.

[20] M. Halstead, M. Kass and T. DeRose, "Efficient, Fair Interpolation using Catmull-Clark Surfaces," *Computer Graphics* (Proc. SIGGRAPH), pp. 35-44, August 1993.

[21] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Mesh Optimization," *Computer Graphics* (Proc. SIGGRAPH), pp. 19-26, August 1993.

[22] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer and W. Stuetzle, "Piecewise Smooth Surface Reconstruction," *Computer Graphics* (Proc. SIGGRAPH), pp. 295-302, July 1994.

[23] D.H. House, D.E. Breen and P.H. Getto, "On the Dynamic Simulation of Physically-Based Particle-System Models," *Third Eurographics Workshop on Animation and Simulation Proceedings* (Cambridge, UK, September 1992).

[24] Y. Hwang and N. Ahuja, “Path Planning Using a Potential Field Representation,” *IEEE Conference on Computer Vision and Pattern Recognition* (IEEE Computer Society Press, Washington, June 1989) pp. 569-575.

[25] M. Kass, A. Witkin and D. Terzopoulos, “Snakes: Active Contour Models,” *International Journal of Computer Vision*, Vol. 1, No. 4, pp. 321-331, January 1988.

[26] A. Kaufman, D. Cohen and R. Yagel, “Volume Graphics,” *Computer*, Vol. 26, No. 7, pp. 51-64, July 1993.

[27] J.K. Kawai, J.S. Painter and M.F. Cohen, “Radioptimization - Goal Based Rendering,” *Computer Graphics* (Proc. SIGGRAPH), pp. 147-154, August 1993.

[28] O. Khatib, ”Real-Time Obstacle Avoidance for Manipulators and Mobil Robots,” *IEEE Conference on Robotics and Automation Proceedings* (IEEE Computer Society Press, Piscataway, NJ, March 1985) pp. 500-505.

[29] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, “Optimization by Simulated Annealing,” *Science*, Vol. 220, No. 4598, pp. 671-680, May 1983.

[30] B.H. Krogh, ”Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles,” *IEEE Conference on Robotics and Automation Proceedings* (IEEE Computer Society Press, Piscataway, NJ, April 1986) pp. 1664-1669.

[31] Z. Liu, S.J. Gortler and M.F. Cohen, “Hierarchical Spacetime Control,” *Computer Graphics* (Proc. SIGGRAPH), pp. 35-42, July 1994.

[32] J. Maillot, H. Yahia and A. Verroust, "Interactive Texture Mapping," *Computer Graphics* (Proc. SIGGRAPH), pp. 27-34, August 1993.

[33] D. MacDonald, D. Avis and A. Evans, "Multiple Surface Identification and Matching in Magnetic Resonance Images," *Visualization in Biomedical Computing '94 Conference Proceedings*, SPIE, Vol. 2359, pp. 160-169, October 1994.

[34] T. McInerney and D. Terzopoulos, "A Dynamic Finite Element Surface Model for Segmentation and Tracking in Multidimensional Medical Images with Application to Cardiac 4D Image Analysis," *Computerized Medical Imaging and Graphics*, Vol. 19, No. 1, pp. 69-83, 1995.

[35] J.V. Miller, D.E. Breen and M.J. Wozny, "Extracting Geometric Models Through Constraint Minimization," *Visualization '90 Conference Proceedings* (San Francisco, CA, October 1990) pp. 74-82.

[36] J.V. Miller, "On GDMs: Geometrically Deformed Models for the Extraction of Closed Shapes from Volume Data," Master's Thesis, RDRC Technical Report TR-90051 (Center for Advanced Technology, Rensselaer Polytechnic Institute, Troy, NY, December 1990).

[37] J.V. Miller, D.E. Breen, W.E. Lorensen, R.M. O'Bara and M.J. Wozny, "Geometrically Deformed Models: A Method of Extracting Closed Geometric Models from Volume Data," *Computer Graphics* (Proc. SIGGRAPH), Vol. 25, No. 4, pp. 217-226, July 1991.

[38] H.P. Moreton and C.H. Sequin, "Functional Optimization for Fair Surface Design," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 167-176, July 1992.

[39] M.E. Mortenson, *Geometric Modeling* (John Wiley & Sons, New York, 1985).

[40] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization* (Addison-Wesley, Reading, MA, 1991).

[41] J.T. Ngo and J. Marks, "Spacetime Constraints Revisited," *Computer Graphics* (Proc. SIGGRAPH), pp. 343-350, August 1993.

[42] H. Okabe, H. Imaoka, T. Tomiha and H. Niwaya, "Three Dimensional Apparel CAD System," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 105-110, July 1992.

[43] C.B. Phillips, J. Zhao and N.I. Badler, "Interactive Real-time Articulated Figure Manipulation Using Multiple Kinematic Constraints," *Computer Graphics* (Proc. Symposium on Interactive 3D Graphics), Vol. 24, No. 2, pp. 245-250, March 1990.

[44] C.B. Phillips and N.I. Badler, "Interactive Behaviors for Bipedal Articulated Figures," *Computer Graphics* (Proc. SIGGRAPH), Vol. 25, No. 4, pp. 359-362, July 1991.

[45] J.C. Platt and A.H. Barr, "Constraint Methods for Flexible Models," *Computer Graphics* (Proc. SIGGRAPH), Vol. 22, No. 4, pp. 279-288, August 1988.

[46] T. Sederberg and E. Greenwood, "A Physically Based Approach to 2-D Shape Blending," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 25-24, July 1992.

[47] T. Sederberg, P. Gao, G. Wang and H. Mu, "2-D Shape Blending: An Intrinsic Solution to the Vertex Path Problem," *Computer Graphics* (Proc. SIGGRAPH), pp. 15-18, August 1993.

[48] J.M. Snyder, "Interval Analysis for Computer Graphics," *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 121-130, July 1992.

[49] J.M. Snyder, A.R. Woodbury, K. Fleischer, B. Currin and A.H. Barr, "Interval Methods for Multi-Point Collisions between Time-Dependent Curved Surfaces," *Computer Graphics* (Proc. SIGGRAPH), pp. 321-334, August 1993.

[50] D. Terzopoulos, A. Witkin and M. Kass, "Symmetry-Seeking Models for 3D Object Reconstruction," *International Journal of Computer Vision*, Vol. 1, No. 3, pp. 211-221, October 1987.

[51] D. Terzopoulos A. Witkin and M. Kass, "Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion," *Artificial Intelligence*, Vol. 36, pp. 91-123, 1988.

[52] D. Terzopoulos and K. Fleischer, "Deformable Models," *The Visual Computer*, Vol. 4, No. 6, pp. 306-331, 1988.

[53] D. Terzopoulos and H. Qin, “Dynamic NURBS with Geometric Constraints for Interactive Sculpting,” *ACM Transactions on Graphics*, Vol. 13, No. 2, pp. 103-136, April 1994.

[54] M. van de Panne and E. Fiume, “Sensor-Actuator Networks,” *Computer Graphics* (Proc. SIGGRAPH), pp. 335-342, August 1993.

[55] J. Weil, “The Synthesis of Cloth Objects,” *Computer Graphics* (Proc. SIGGRAPH), Vol. 20, No. 4, pp. 359-376, August 1986.

[56] W. Welch and A. Witkin, “Variational Surface Modeling,” *Computer Graphics* (Proc. SIGGRAPH), Vol. 26, No. 2, pp. 157-166, July 1992.

[57] W. Welch and A. Witkin, “Free-Form Shape Design Using Triangulated Surfaces,” *Computer Graphics* (Proc. SIGGRAPH), pp. 247-256, July 1994.

[58] R.T. Whitaker and D.T. Chen, “Embedded Active Surfaces for Volume Visualization,” *Medical Imaging '94: Image Processing Conference Proceedings*, SPIE, Vol. 2167, pp. 340-352, February 1994.

[59] R.T. Whitaker, “Volumetric Deformable Models: Active Blobs,” *Visualization in Biomedical Computing '94 Conference Proceedings*, SPIE, Vol. 2359, pp. 122-134, October 1994.

[60] A. Witkin, K. Fleischer and A. Barr, “Energy Constraints on Parameterized Models,” *Computer Graphics* (Proc. SIGGRAPH), Vol. 21, No. 4, pp. 225-232, July 1987.

[61] A. Witkin and M. Kass, "Spacetime Constraints," *Computer Graphics* (Proc. SIGGRAPH), Vol. 22, No. 4, pp. 159-168, August 1988.

[62] A.P. Witkin and P.S. Heckbert, "Using Particles to Sample and Control Implicit Surfaces," *Computer Graphics* (Proc. SIGGRAPH), pp. 269-277, July 1994.