Interactive Haptic Rendering of Deformable Surfaces Based on the Medial Axis Transform

Jason J. Corso, Jatin Chhugani and Allison M. Okamura
The Johns Hopkins University
3400 North Charles Street
Baltimore, Maryland, 21218, USA
{jcorso,jatinch,aokamura}@jhu.edu

Abstract

We present a new method for interactive deformation and haptic rendering of viscoelastic surfaces. Objects are defined by a discretized Medial Axis Transform (MAT), which consists of an ordered set of circles (in 2D) or spheres (in 3D) whose centers are connected by a skeleton. The skeleton and physical properties of the object, including the radii of the spheres centered on the skeleton and material properties, are encapsulated in a single high dimensional parametric surface. To compute the force upon deformation, we use a mass-spring-damper model that takes into account both normal and shear surface forces. Our implementation attains real time 3D haptic and graphic rendering rates, making it appropriate to model deformation in complex haptic virtual environments. The algorithm is appealing because it takes advantage of single-point haptic interaction to render efficiently while maintaining a very low memory footprint.

1. Introduction

Physical objects in the real world possess attributes far beyond those present in typical virtual reality systems. Visual, haptic, aural, taste, and smell attributes are required to fully describe an object. Incorporating deformable viscoelastic surfaces into virtual environments involves computationally intensive tasks.

Many deformable surfaces are by nature smooth, and their visual rendering must yield an accurate representation on the display. This graphic rendering must also be performed at fast rates (10-20Hz). These two demands must be balanced to produce fast, but accurate graphical rendering of the surfaces. To enable real-time graphic rendering rates, specialized hardware has been developed that quickly rasterizes triangles numbering on the order of 10^6 per frame. Triangles, however, are discrete and do not enable view-dependent adaptation for varying levels of resolution. Parametric surfaces provide a means to render these viscoelastic surfaces at varying resolutions. The parametric surfaces undergo a process called tessellation during which a set of triangles are created to approximate the smooth surface; the set of triangles created is bound by either object space error or screen space error allowing for arbitrary levels of resolution.

Resolved-force haptic interaction mandates an intersection calculation of the user's position with the surfaces at haptic interactive rates (1 kHz). Single-point collision calculation with implicit surfaces involves a simple sign evaluation of an expression describing the surfaces.

However, the needs for efficient graphical rendering contradict those for good haptic rendering. While implicit surfaces are superb for use in haptic rendering, their graphical rendering is extremely slow. And while parametric surfaces are the state-of-the-art in graphics rendering, employing them in a haptic environment yields more complicated resolved-force calculations resulting in a slower system. In this work, we extend the current state-of-the-art virtual environment technology to include a new method for combined real-time visual and resolved-force haptic interaction with deformable surfaces.

We continue this introductory section by describing the previous and related work. The algorithm for surface modeling is then explained in Section 2. Object intersection and deformation is presented in Sections 3.1 and 3.2. We discuss our approach to haptic and graphic rendering in Sections 3.4 and 3.5. Section 4 describes the implementation and results. Finally, conclusions are drawn and future directions are listed in Section 5.

1.1 Previous / Related Work

The foundation of our algorithm is the use of shape skeletons as a basis for the object model. Shape skeletons

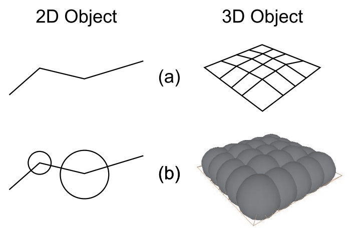

Figure 1. Development of an object from the Medial Axis Transform includes: (a) the original skeleton, (b) the original skeleton with the circles/spheres, and (c) the spline approximation to the skeleton and surface.

are geometric abstractions of curves or surfaces that are useful as lower-dimensional representations. The skeleton is known in 2D as the *medial axis* and in 3D as the *medial surface*. Each point on the medial axis or surface is associated with the radius of a locally maximal disk or sphere. These medial points, together with their associated radii, define the MAT of an object. An example of a 2D/3D medial axis/surface with the associated circles/spheres is shown in Figure 1. The MAT was first proposed by Blum [4] as an alternative shape description for biological applications. It has recently been used by Pizer et al. [22] as a multilocal and multiscale representation for graphic and computer-aided design applications, where a figure is defined by a mesh of medial "atoms." Gagvani [9] has recently employed skeletonization to automatically generate the volumetric representation of a polygonal mesh. He explores the approach for use in volumetric modeling, deformation, and animation. Okamura et al. [19] developed a robotic system to acquire medial axis-based models of rigid surface features on real 3D objects. Their work demonstrates the usefulness of the MAT in modeling real-world objects.

Finite Element Methods (FEM) and Boundary Element Methods (BEM) [11] have been used for physical representation in haptic environments. FEM and BEM, while being computationally and memory expensive, can accurately compute deformation and contact forces. The work of Pai et al. [12, 13] (called ArtDEFO) employs BEM for accurate deformation; it exploits the coherence of typical physical interaction in order to achieve interactive rates for relatively small models. FEM/BEM have been used in the medical domain for surgery simulation [3, 6, 17]. These simulations include the ability to dynamically cut and tear the model.

In the graphics community, object deformation has been successfully implemented using many types of object models. However, most of these methods suffer in application to haptic rendering because they cannot provide force feedback at interactive rates. We refer to the technical report by Gibson et al. [10] for a survey of deformable models used in computer graphics. Free-form solid deformation [5, 23] is the most broad approach to modeling real solids. It has been used as an extension of Geometric Modeling to create more realistic design environments for human modelers; it separates the underlying geometry from the model as much as possible. Sensable Technologies Inc. (Woburn, MA) has developed a product called the FreeForm^{TM}Modeling System. Its existence demonstrates an industry wide need for haptic-based modeling environments. Volumetric approaches have also been taken to model deformable objects. Avila and Sobierajski [1] present a volumetric method suitable for both visualization and modeling applications. They compute point contact forces directly from the volume data consistent with isosurface and volumetric graphic rendering methods, thereby ensuring consistency between haptic and visual feedback. In more recent work, Frisken and Perry [8] have explored the use of Adaptively Sampled Distance Fields to model soft body contact in a computationally efficient framework.

NURBS surfaces have also been used in haptics-based applications because of their dominance in the CAD/CAM fields. Object modeled as NURBS surfaces can be rendered interactively at varying levels of resolution with bounded error [15]. This makes them attractive for use in graphically intensive applications. Another set of algorithms [14, 18, 25] have been developed for direct haptic rendering of NURBS based models, including an extension to surface-surface interaction which removes the single-point of intersection constraint commonly found in haptic rendering. Terzopoulos and Qin [24] have developed D-NURBS, a physics based generalization of NURBS curves and surfaces. D-NURBS are derived through the application of Lagrangian mechanics and implemented using FEM.

In our work, we present a novel approach that employs an extended MAT to compute haptic interaction complemented by the NURBS representation to render visually appealing surfaces. It integrates the needs of good haptic and graphic rendering by combining a parametric skeleton encapsulating an underlying set of implicit shapes (circles in 2D and spheres in 3D) with a parametric surface for accurate view-dependent graphic rendering. The salient features of our method are (**1**) object models are easily created using a discretized Medial Axis Transform (MAT), and the algorithm automatically generates a high dimensional parametric surface that encapsulates object shape, stiffness, damping, and mass, (**2**) the algorithm for collision detection and haptic feedback of normal and surface shear forces is efficient, taking advantage of single-point haptic interaction, and (**3**) the addition of dynamic behavior is straightforward

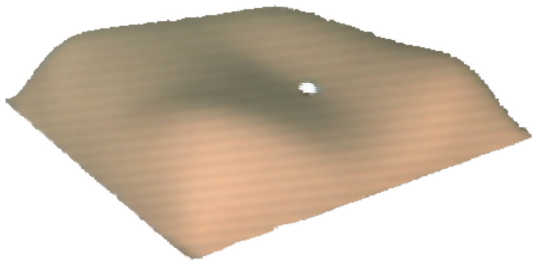

Figure 2. This dynamically deformable 3-dimensional object is haptically rendered at 1kHz on a standard PC. The white dot represents the location of the haptic interface in the virtual environment.

because deformation is directly calculated from object properties included in the parametric surface. Our implementation, *DeforMAT*, achieves real time rendering rates for haptic interaction, which are on the order of 1 kHz. An image of a virtual environment with a MAT object is shown in Figure 2.

2. Object Modeling

An object (Figure 1) consists of the following attributes:

1. The discretized medial surface (skeleton), obtained using a valid MAT on the original surface of a real body. A number of algorithms [4, 7, 9] have been proposed which can extract the skeleton from the description of any given body.
2. Radii of the spheres centered along the skeleton.
3. Stiffness (and potentially other material properties) of each sphere centered along the skeleton.

From the above given data, we construct the following attributes:

1. Skeleton Spline (**SK**), a high dimensional Open, Non-Uniform B-Spline Surface[1] that fits the position, radius, and material properties associated with each skeleton point.

2. Surface Contour (**SC**), a 3D Open, Non-Uniform B-Spline Surface that is the explicit body contour. This is used only in graphic rendering.

When the user interacts with bodies in the virtual environment, we determine whether the user is intersecting a body using the medial surface **SK** (Section 3.1). This is followed by computing the dynamic deformation of the body (Section 3.2).

Two limitations of this modeling technique are that points on the medial surface must be ordered and that the surface cannot bifurcate. Recently, Dey et al. [7] have proposed an algorithm for computing an ordered set of points on the medial axis from a set of surface points. This algorithm can be used to generate the regular grid of medial axis points as the input for our algorithm.

For the remainder of the paper we will consider 3D objects; simplification to 2D is straightforward.

2.1. Surface Modeling

The input is given in the form of a regular grid of 3D points on the medial surface (skeleton), $\mathbf{P_{1,1}}...\mathbf{P_{1,n}}...\mathbf{P_{m,n}}$, as well as the radii of the spheres, $r_{1,1}...r_{1,n}...r_{m,n}$, and surface stiffness, $k_{1,1}...k_{1,n}...k_{m,n}$, corresponding to these points. One can also vary the surface damping and mass for dynamic deformations, but we will omit these here for clarity. The surface modeling consists of interpolating a 5-dimensional B-Spline Surface through the skeleton points, radii and stiffness values. Moreover, for each given point on the skeleton and the radius of the sphere at that point, the corresponding point on the body surface is obtained by adding the radius along the medial surface normal from the medial surface point. Hence, after we compute a B-Spline Surface through the skeleton **SK**, we compute points on the object surface and interpolate those points to get the boundary surfaces **SC** for both the top and the bottom half of the body. One can incorporate these values in the skeleton (**SK**) itself, but this leads to a quadratic increase in the degree of the resulting surface. The detailed algorithm is as follows:

1. Interpolate a 3D B-Spline Surface (henceforth referred to as **SK**) through the $m*n$ data points on the skeleton to obtain a medial surface with parameters u and v. We employ the chord-length parameterization method for knot selection, and an efficient implementation for a linear system solution that computes the control points: we invoke MATLAB®from inside our program to perform the linear system solution. Let the minimum and maximum knot values be U_{min}, V_{min} and U_{max}, V_{max} respectively.
2. Model the radii of the spheres as another B-Spline Surface, and add the control points as an extra dimension to **SK**, making it into a 4D surface. Similarly, add the stiffness as another dimension, creating a 5D surface. Damping and mass can be added as well to simulate a second order dynamic system.
3. Given the new, high dimensional surface **SK**, compute the derivatives of this surface (just the three spatial dimensions) to obtain the derivative surfaces $\mathbf{SKD_u}$ and $\mathbf{SKD_v}$, which are 3D B-Spline Surfaces.

[1]For a detailed treatment of B-Spline Curves and Surfaces, the reader is referred to [2].

4. For $i \in [1..m]$ and $j \in [1..n]$:
   - For the given point $\mathbf{P_{i,j}}$ on the medial surface, compute the normal (using $\mathbf{SKD_u}$ and $\mathbf{SKD_v}$), $\mathbf{N_{i,j}}$.
   - Obtain $\mathbf{R_{i,j}} = \mathbf{P_{i,j}} + r_{i,j} * \mathbf{N_{i,j}}$ and $\mathbf{S_{i,j}} = \mathbf{P_{i,j}} - r_{i,j} * \mathbf{N_{i,j}}$ where $r_{i,j}$ is the radius of the circle at $\mathbf{P_{i,j}}$. To ensure surface continuity, $r_{i,j}$ is assumed to be equal to zero at the end points.
5. Interpolate an Open, Non-Uniform 3D B-Spline Surface through $\mathbf{R_{i,j}}, i \in [1..m], j \in [1..n]$ to obtain the surface for the top part of the body, and through $\mathbf{S_{i,j}}, i \in [1..m], j \in [1..n]$ to obtain the surface for the bottom part of the body. These two surfaces together form the body contour, **SC**.

3. Object Interaction

3.1. Intersection Computation

We perform an efficient intersection computation which gives us the position of the user relative to boundary of the surface within sub-pixel accuracy.[2] We execute a binary search over the knot value range in the two directions to determine the parametric values, such that the normal drawn on the skeleton spline **SK** passes through the point on which the user lies. Let (u_{begin}, v_{begin}) and (u_{end}, v_{end}) represent the two end points of the interval being examined (Figure 3). Let **Q** represent the position of the user. The algorithm terminates when the length of the interval (in both the directions) falls below pre-defined thresholds, or when the normal passes through the point **Q**. The algorithm returns the parameter value, say $(u_{intersect}, v_{intersect})$, such that the normal drawn on the **SK** at $(u_{intersect}, v_{intersect})$ (call this point **P**) would pass through **Q**.

Now compute the distance between **P** and **Q** ($||\mathbf{PQ}||$), and radius of the circle at $(u_{intersect}, v_{intersect})$ (called $R_{intersect}$). If ($||\mathbf{PQ}|| <= R_{intersect}$) then the user intersects the object, otherwise, the user is outside the object. The algorithm terminates in maximum of $1 + log_2 \frac{1}{u_{threshold}}$ steps. A value of $u_{threshold} = v_{threshold} = 2^{-10}$ works well in practice.

It is important that this algorithm be efficient because it must be executed every haptic loop, i.e. approximately 1000 times a second.

3.2. Object Deformation

After detecting that the user is inside the object using the algorithm described above, we need to change the object visually and apply forces on the user. Let the user be penetrating a distance of δ inside the boundary of the object, where

$$\delta = R_{intersect} - ||\mathbf{PQ}||.$$

To reflect the deformation of the body, we need to change the skeleton spline (**SK**), including the radii of the circles in the neighborhood of the region, and the contour spline (**SC**) (Figure 4).

One can consider two schemes for body deformation. In the first scheme, the interaction moves the skeletal surface **SK**, and both the body surfaces follow. In the second scheme, only the body surface intersecting with the user (e.g., the top surface) moves, while the other surface (e.g., the bottom surface) does not. To model this type of body compression, let a skeleton point be displaced by a distance of $g\delta$, while the radii spline is displaced by $(1-g)\delta$. When $g = 0.5$, the contour of the body opposite the user is not displaced. The algorithm for displacing the splines given δ and **PQ** is explained later in Section 3.3.

In both schemes, the object can be deformed dynamically through the use of a second order dynamic system.

[2] An iterative solver is not suited for real-time haptic interactions. The current implementation is only guaranteed to work for convex medial surfaces. In practice, errors occur rarely with concave medial surfaces.

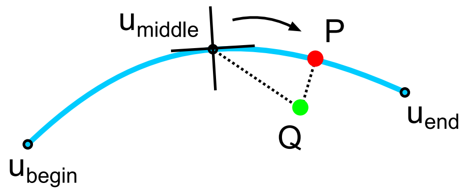

Figure 3. Intersection computation involves iteratively modifying the values of parameters u and v to obtain the point P on the spline perpendicular to the position of the user, point Q. For clarity, this figure shows only one parameter, u.

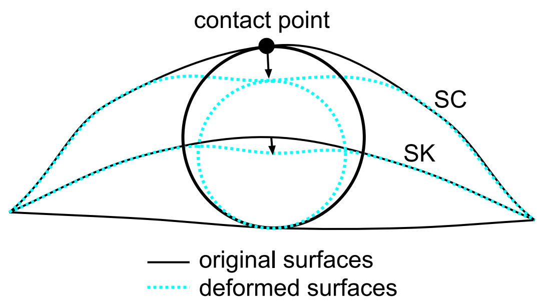

Figure 4. Displacement of the object skeleton and contour upon contact, shown in 2D.

This approach models organic bodies well because of their damped elastic, non-rigid nature. We imagine springs and dampers at the control points of the contour spline (**SC**) and skeleton spline (**SK**). When the user interacts with the model, these springs undergo a change in length, and an internal force is felt by the control points, pulling them back towards their original positions. Dampers are added to reduce the oscillation of the surface.

The positions of the control points are evaluated using standard second order dynamic equations. Let p_{prev}, v_{prev} and a_{prev} be the position, velocity and acceleration respectively of a control point in the previous frame. Let p_{neut} be the original position of that control point. The current force on the control points is

$$f = -k * (p_{prev} - p_{neut}) - b * (v_{prev}),$$

where k is the stiffness coefficient of the spring and b is the damping coefficient. The current acceleration of the control point is $a_{curr} = \frac{f}{m}$, where m is the simulated mass of the control point. Using a_{curr}, the current velocity (v_{curr}) and position (p_{curr}) are computed. To estimate appropriate values for b and m, we observe the dynamic equation:

$$m\frac{d^2\mathbf{x}}{dt^2} + b\frac{d\mathbf{x}}{dt} + k\mathbf{x} = 0,$$

where $\mathbf{x}$ is the displacement vector of the mass, and t is the time. For $(b^2 - 4km) < 0$, the displacement vector undergoes damped sinusoid oscillations, with the amplitude varying as $e^{-\frac{bt}{2m}}$. Hence, the values of b, m and k are chosen so that the points come to rest in a short time. The algorithm for displacing the splines is explained in the following section.

3.3. Spline Deformation

Here we describe the approach we have taken to displace the control points of the body surface **SC** given the displacement vector of a point on the surface. Commonly used techniques include changing the knot vector and the weights of the control points (for rational B-Splines) [20, 21]. The technique we use is an extension of Piegl's control point modification scheme. Our method involves modifying the $p' * q'$ most influenced control points in proportion to their contribution to the contact point on the surface, and $p' \leq p+1$, $q' \leq q+1$, where p and q are the degrees of the surface in the u and v directions respectively. (The values for p' and q' should be set with symmetry, such that if p is odd, then p' should be even and vice versa). This guarantees a smooth change in the shape of the modified B-Spline Surface as the set of control points change with the displacement of the curve. For a better approximation, one can modify up to $(p+1)*(q+1)$ control points. However, in practice, for a surface of degree (p,q), displacing $\frac{p}{2} * \frac{q}{2}$ control points gives a visually acceptable approximation. In order to obtain real-time frame rates, the number of changed points should be kept to a minimum. Because deformations move multiple control points, we retessellate the entire surface at each time step.

For explanation purposes, assume a 2D B-Spline Curve of degree 4 with control points $\mathbf{C}_i$, $i \in [1..n]$. A point $\mathbf{P}$, having a parametric value t, has to be moved by a distance d along the vector $\hat{\mathbf{v}}$. Say the parametric value t lies in the knot vector range $[t_j, t_{j+1})$. The control points affecting any parameter value in this range are $[\mathbf{C}_j..\mathbf{C}_{j-4}]$. We choose to deform only $\mathbf{C}_{j-1}$, $\mathbf{C}_{j-2}$ and $\mathbf{C}_{j-3}$. Let the basis values for these control points be $B_1(t)$, $B_2(t)$ and $B_3(t)$. Let these control points be displaced by $\alpha_1\hat{\mathbf{v}}$, $\alpha_2\hat{\mathbf{v}}$ and $\alpha_3\hat{\mathbf{v}}$. Hence,

$$d = \alpha_1 B_1(t) + \alpha_2 B_2(t) + \alpha_3 B_3(t).$$

Also $\alpha_1 : \alpha_2 : \alpha_3 = B_1(t) : B_2(t) : B_3(t)$. These two equations imply:

$$\mathbf{C}_{j-1} = \mathbf{C}_{j-1} + \frac{dB_1(t)}{B_1^2(t) + B_2^2(t) + B_3^2(t)}\hat{\mathbf{v}}$$

$$\mathbf{C}_{j-2} = \mathbf{C}_{j-2} + \frac{dB_2(t)}{B_1^2(t) + B_2^2(t) + B_3^2(t)}\hat{\mathbf{v}}$$

$$\mathbf{C}_{j-3} = \mathbf{C}_{j-3} + \frac{dB_3(t)}{B_1^2(t) + B_2^2(t) + B_3^2(t)}\hat{\mathbf{v}}.$$

In the more general 3D case, given a B-Spline Surface of degree (p,q) with control points $\mathbf{C}_{i,j}$, $i \in [1..m]$, $j \in [1..n]$, we have demonstrated that deforming the $p' * q'$ grid of most influential control points achieves the necessary deformation. Assume that a point $\mathbf{P}$, having a parametric value (s,t), has to be moved by a distance d along the vector $\hat{\mathbf{v}}$. Let the parametric values s and t lie in the knot vector ranges $[s_k, s_{k+1})$ and $[t_l, t_{l+1})$ respectively. The control points affecting any parameter value in this range are $[\mathbf{C}_{k,l}...\mathbf{C}_{k-p,l-q}]$. We choose to deform only $\mathbf{C}_{k-\frac{p+1+p'}{2}, l-\frac{q+1+q'}{2}}...\mathbf{C}_{k-\frac{p+p'-1}{2}, l-\frac{q+q'-1}{2}}$ using equations similar to those above. Hence, we can modify the control points to reflect the displacement of the B-Spline Surface.

3.4. Haptic Rendering

We use a second-order mass-spring-damper system to compute the force exerted on the user as he or she tries to penetrate into the body of the object. Let the user penetrate a distance x into the contour surface at a position corresponding to the parameter value (u,v). The spring centered on the skeleton at parameter value (u,v) is compressed by a distance x, and the applied force (using Hooke's Law) is

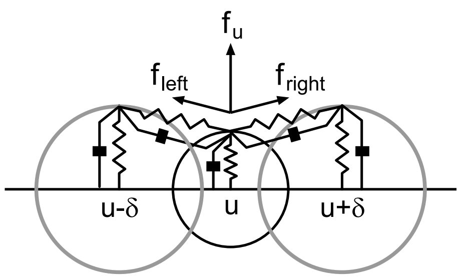

Figure 5. Forces applied to the user upon contact are calculated from a second-order spring system. One spring, connecting the contact point to the skeleton, pushes the user in a direction normal the surface. Four springs (only two are shown in this 2D view) apply shear force from nearby points on the surface contour.

$k_{u,v}x$, where $k_{u,v}$ is the stiffness value obtained from **SK**. (The damping $b_{u,v}$ and mass $m_{u,v}$ are also obtained from **SK**.) To model the shear forces along the surface (as a result of the strain on the contour), we consider four more spheres, at a distance δ (in parametric space in both the u and v directions), on either side of the point of intersection. These springs are henceforth referred to as $K_{u-\delta}$, $K_{u+\delta}$, $K_{v-\delta}$ and $K_{v+\delta}$. The resultant spring system in one parametric direction is shown in Figure 5 . A similar placement would hold for the other parametric axis. Let the displacement of the springs be $x_{u-\delta}$, $x_{u+\delta}$, $x_{v-\delta}$ and $x_{v+\delta}$. The stiffness values of the springs are set to $keq_{u-\delta} = \frac{k_{u-\delta,v}+k_{u,v}}{2}$, $keq_{u+\delta} = \frac{k_{u+\delta,v}+k_{u,v}}{2}$, $keq_{v-\delta} = \frac{k_{u,v-\delta}+k_{u,v}}{2}$ and $keq_{v+\delta} = \frac{k_{u,v+\delta}+k_{u,v}}{2}$. The net force $f_{feedback}$ on the user is equal to $k_{u,v} * x + keq_{u-\delta} * x_{u-\delta} + keq_{u+\delta} * x_{u+\delta} + keq_{v-\delta} * x_{v-\delta} + keq_{v+\delta} * x_{v+\delta}$. It can be shown that the force always increases as the user penetrates the object, and the direction of the feedback force always pushes the user out of the object.

3.5. Graphic Rendering

We used the OpenGL API to display the surfaces. To render the virtual objects, we used the GLU NURBS Tessellator which computes the triangulation of every surface (within a user specified screen-space error), and renders these triangles to compose the final image. This tessellation is done every frame, with frame rates varying between 10-20 frames for most of the models. We also incorporate basic view frustum culling to further speed up the rendering rates. One of the major advantages of tessellating a surface for rendering is that it exploits the fast triangle rendering capability of the hardware, thereby leading to high graphic update rates. It is important to note that the complexity of the virtual environment being graphically rendered is on an average about 10^5 triangles and is independent of the number of spheres because of the view-dependent adaptive tessellation of the NURBS surface.

4. Implementation and Results

Our system is developed in C++ on a standard 700MHz Pentium III Computer with 384 MB of main memory. Virtual world navigation is performed using the computer mouse, and for force feedback we employ a PHANTOM^{TM}Premium 1.5 3-degree-of-freedom haptic device from SensAble Technologies, Inc. (Woburn, MA). For graphics hardware, we used the NVidia GeForce2$^{©}$ card. The haptic device control ran in a separate high-priority thread at 1kHz. Table 1 shows the performance of our algorithm with objects of varying complexity.

5. Conclusion and Future Work

We present a new algorithm for interactively deforming viscoelastic bodies at haptic interactive rates, i.e. 1 kHz. Our system fits a niche much needed in the virtual environment arena: namely, the ability to add dynamic deformations coupled with haptic feedback in a virtual environment with minimal cost. The algorithm balances the contradictory demands of haptic rendering with those of graphic rendering in a manner well suited for numerous applications, including medical simulation, art, and entertainment. One example of a medical application is a training system for tumor location using palpation.

This initial implementation of our algorithm includes the minimum features required to simulate dynamic, deformable surfaces. There exist many avenues for future work, including bifurcating and non-ordered medial axes/surfaces, analysis of area/volume preservation, implementation of more efficient graphical rendering algorithms ([15, 16, 22]), experimentation with different deformation modes, and direct performance comparison against other methods.

Acknowledgments

We acknowledge Samuel Khor for starting the work on haptic rendering using shape skeletons at The Johns Hopkins, and thank Budirijanto Purnomo for his assistance with B-Spline deformation.

# Bodies	# MAT Points	Preprocessing Time	Memory	Rate (w/Dynamics)		Rate (w/o Dynamics)		Time in Haptics Loop
				Haptics	Graphics	Haptics	Graphics	
1	49	2 sec	400 KB	980 Hz	17 Hz	993 Hz	20 Hz	20%
1	105	3 sec	1.0 MB	979 Hz	14 Hz	996 Hz	18 Hz	26%
4	213	6 sec	1.8 MB	990 Hz	11 Hz	993 Hz	15 Hz	25%
10	309	9 sec	2.1 MB	980 Hz	10 Hz	992 Hz	17 Hz	27%
17	465	25 sec	3.0 MB	975 Hz	9 Hz	980 Hz	12 Hz	30%
35	927	56 sec	4.1 MB	980 Hz	4-5 Hz	990 Hz	6 Hz	36%

Table 1. Our algorithm's performance for virtual environments of varying complexity. Listed are the number of bodies in the virtual environment, the total number of MAT points in all the bodies, the preprocessing time (used to fit splines to the MAT), memory required, update rates with and without dynamic calculations for both the haptics and graphics loops, and the percent of run time spent in the haptics thread.

References

[1] R. Avila and L. Sobierajski. A haptic interaction method for volume visualization. *Proc. of IEEE Visualization '96*, pages 197–204, 1996.

[2] R. Bartels, J. Beatty, and B. Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufman, 1987.

[3] D. Bielser and M. H. Gross. Open surgery simulation. In *Proc. of Medicine Meets Virtual Reality*, 2002.

[4] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. M.I.T. Press, Cambridge, MA, 1967.

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

[6] S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999.

[7] T. K. Dey and W. Zhao. Approximate medial axis as a voronoi subcomplex. In *Proceedings of 7th Symposium on Solid Modeling Applications*, 2002 *to appear*.

[8] S. Frisken and R. Perry. A computationally efficient framework for modeling soft body impact. Technical Report TR2001-11, MERL, 2001.

[9] N. Gagvani. *Parameter-Controlled Skeletonization - A Framework for Volume Graphics*. PhD thesis, Rutgers, The State University of New Jersey, 2000.

[10] S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR97-19, MERL, 1997, http://www.merl.com/papers/docs/TR97-19.pdf.

[11] P. Hunter and A. Pullan. Fem bem notes. Technical report, University of Auckland, 1998, http://www.esc.auckland.ac.nz/Academic/Texts/FEM-BEM-notes.html.

[12] D. L. James and D. K. Pai. Artdefo - accurate real time deformable objects. *Siggraph 1999, Computer Graphics Proc.*, pages 65–72, 1999.

[13] D. L. James and D. K. Pai. A unified treatment of elastostatic contact simulation for real-time haptics. *haptics-e.org*, 2(1), 2001.

[14] D. Johnson and E. Cohen. An improved method for haptic tracing of sculptured surfaces, 1998.

[15] S. Kumar, D. Manocha, and A. Lastra. Interactive display of large-scale nurbs models. *Proc. of the 1995 Symposium on Interactive 3D Graphics*, pages 51–58, 1995.

[16] F. Li, R. Lau, and M. Green. Interactive rendering of deforming nurbs surfaces. *Computer Graphics Forum*, 16(3):47–56, 1997.

[17] C. Mendoza, C. Laugier, and F. Boux de Casson. Towards a realistic medical simulator using virtual environments and haptic interaction. In *Proc. of the International Symposium in Research Robotics*, Lorne, Victoria (AU), 2001.

[18] D. Nelson, D. Johnson, and E. Cohen. Haptic rendering of surface-to-surface sculpted model interaction, 1999, http://citeseer.nj.nec.com/nelson99haptic.html.

[19] A. M. Okamura and M. R. Cutkosky. Feature-guided exploration with a robotic finger. *Proc. of the 2001 IEEE International Conference on Robotics and Automation*, 1:589–596, 2001.

[20] L. Piegl. Modifying the shape of rational b-splines. part 1: Curves. *Computer-Aided Design*, 21(8):509–518, 1989.

[21] L. Piegl. Modifying the shape of rational b-splines. part 2: Surfaces. *Computer-Aided Design*, 21(9):538–546, 1989.

[22] S. M. Pizer, A. L. Thall, and D. T. Chen. M-reps: A new object representation for graphics. Technical Report TR99-030, University of North Carolina, 1999, http://www.cs.unc.edu/Research/Image/MIDAG/pubs/papers/mreps-2000/mrep-pizer.PDF.

[23] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. *Computer Graphics*, 20(4):151–160, 1986.

[24] D. Terzopoulos and H. Qin. Dynamic NURBS with geometric constraints for interactive sculpting. *ACM Transactions on Graphics*, 13(2):103–136, 1994.

[25] T. V. Thompson II, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Symposium on Interactive 3D Graphics*, pages 167–176, 1997.