Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Interactive rendering of deformable objects based on a filling sphere modeling approach

François Conti
conti@robotics.stanford.edu

Oussama Khatib
ok@robotics.stanford.edu

Charles Baur
charles.baur@epfl.ch

Robotics Laboratory
Computer Science Department
Stanford University
Stanford, California 94305, USA

VRAI Group
IPR - LSRO
EPFL
CH - 1015 Lausanne, Switzerland

Abstract

Mass-spring systems have widely and effectively been used for modeling in real-time deformable objects. Easier to implement and faster than finite elements, these systems, on the other side, suffer from several drawbacks when coming to render physically believable behaviors. Neither isotropic or anisotropic materials can be controlled easily and the large number of springs and mass points composing the model makes it fastidious to define parameters to control elongation, flexion and torsion at a macroscopic level. Another weakness is that most of the materials found in nature maintain a constant or quasi-constant volume during deformations; unfortunately, mass-spring models do not have this property.

In this paper, we extend the current state-of-the-art in soft tissue simulation by introducing a six-degree of freedom macroscopic elastic sphere described by mass, inertia and volumetric properties. Spheres are placed along the medial axis transform of the object whose centers are connected by a skeleton composed of a set of three-dimensional elastic links. Spheres represent internal mass, volume and control the global deformation of the object. The surface is modeled by setting point masses on the mesh nodes and damped springs on the mesh edges. These nodes are connected to the skeleton by individual elastic links, which control volume conservation and transfer forces between the surface and volumetric model. Using this framework we also present an efficient method to approximate collision detection between multiple bodies in real-time.

1 Introduction

Rendering deformable objects in a realistic manner is a challenging task in computer simulation. While rigid bodies can be accurately described with a limited number of parameters, deformable objects, on the other side, due their internal structure, require a much larger set of numbers to express their state. During these last two decades, a wide variety of approaches have been presented in soft tissue simulation. Mass-spring system techniques have widely and effectively been used for modeling deformable objects where described by a set of mass points dispersed throughout the object and interconnected with each other through a network of springs. These systems are easy to model, to construct and have well understood physics. They are also well suited for parallel computation, making it possible to run complex environments in real-time for interactive simulations. (A surgery simulator for instance). On the other side, mass-spring systems have some drawbacks. Incompressible volumetric objects and high stiffness materials have poor stability requiring small time steps during the numerical integration process, which considerably slows down the simulation. A second category of techniques is finite elements methods, which offer a method with much higher accuracy to solve continuum models. In FEM, unlike mass-spring methods where the equilibrium equation is discretized and solved at each finite mass point, objects are divided into unitary surface (2D) or volumetric (3D) elements joined at discrete node points where a continuous equilibrium equation is approximated over each element. While finite element methods generate a more physically realistic behavior, at the same time they require much more numerical computation and therefore are difficult to use for real-time simulations.

Designing an interactive haptic simulator is a difficult challenge. If physical accuracy is a key factor towards rendering physically believable scenes, nevertheless, real-time control remains indispensable to produce interactive simulation; users will be disturbed if latencies or interrupts are introduced to the user input.

In this work, we extend the state-of-the-art in mass-spring concepts, by proposing an alternative method to model deformable objects. This method generates and connects together both a surface and volumetric representation based on the medial axis skeleton of a solid. This new algorithm is appealing because it decouples local and global deformation and renders them together with variable levels of resolution.

The paper is organized as follows. In paragraph 2, we introduce the different aspects and details of the algorithm. Collisions between multiple bodies and point-contact haptic interaction are discussed in paragraphs 3 and 4. Implementation of our system and experimental results are showed in paragraph 5 and 6 and finally a description of our future work and a conclusion are presented in paragraph 7.

2 Object Modelling

Given a closed surface, defined by a set of triangle, we construct the following attributes:

- A medial axis skeleton, obtained by using a valid MAT on the original surface of the object. (Figure 1a).
- A volumetric model, composed of filling spheres placed along the medial axis skeleton and inter-connected together via three-dimensional elastic links. (Figure 1b).
- A surface model, realized by setting point masses on the mesh nodes and damped springs on the mesh edges. Nodes are connected to the volumetric model (skeleton composed of spheres and links) through individual elastic links. (Figure 1c).

The following paragraphs (2.1, 2.2 and 2.3) describe in detail the algorithms developed for each stage. Figure 1 illustrates the procedure on a 2D object. For the remainder of the paper we will consider 3D objects.

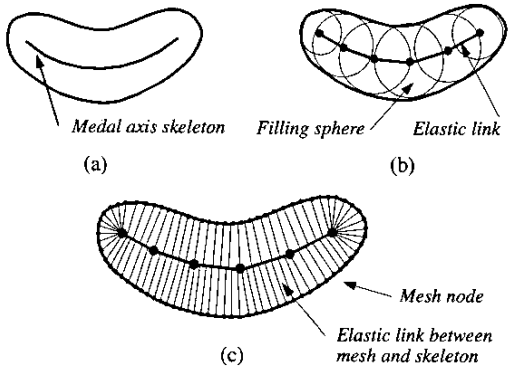

Figure 1 - Medial Axis Skeleton and its corresponding surface (a). Volumetric model composed of spheres and links centered along the skeleton (b). Connections between surfaces nodes and skeleton (c)

2.1 Medial axis transform

The notion of skeleton was introduced by H. Blum [2] as a result of the Medial Axis Transform (MAT) or Symmetry Axis Transform (SAT). The MAT determines the closest boundary points for each point in an object. An inner point belongs to the skeleton if it has at least two closest boundary points. A very illustrative definition of the skeleton is given by the prairie-fire analogy: the boundary of an object is set on fire and the skeleton is the loci where the fire fronts meet and quench each other.

To approximate a skeleton we implemented a method presented by Li and Al. [7] based on an edge-collapsing algorithm. At every round the triangle edge *(u,v)* with shortest Euclidean distance of the mesh and any associated faces collapse into a point at the average location of its endpoints *u* and *v*, and triangles incident to the edge are removed. During the process, whenever an edge *(u,v)* is not incident to any triangle, it is designated as a skeletal edge and vertices u and v maintain their positions until the end of the process. The process iterates until all triangles have been collapsed to edges or vertices. In other words, we are left with only skeletal edges, whose union is the skeleton of the mesh. This method was selected among other skeletonization techniques for its performance in speed and ease of implementation with surface based objects. In figure 2, we illustrate a clapsing edge example based on a 466 triangles mesh representing a gallbladder; the generated skeleton is composed of seven nodes and six links. By using a Pentium III - 1GHz computer, the complete skeletonization process took 1.5 seconds.

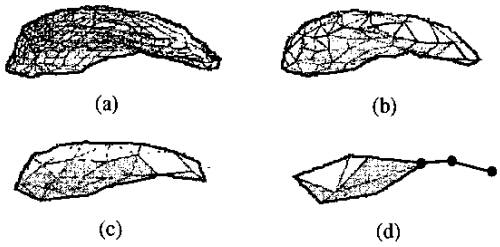

Figure 2 - Skeletonization of a triangle based mesh using a clapsing edge algorithm. The original mesh is composed of 466 triangles and collapses into a skeleton composed of 7 nodes and 6 links

2.2 Volume Modeling

A set of filling spheres compose the volumetric model. Spheres are placed inside the object, along the medial axis transform, and are linked together via elastic links. A sphere represents a portion of mass of an object and is described by a reference frame *R*, mass *m* and radius *r*. Spheres can freely translate and rotate in space unless constraints are applied; for instance, by constraining translation on certain spheres, it is possible to attach deformable bodies to ground points. Elastic links, connecting adjacent spheres together, are composed of six damped springs controlling *elongation, flexion* and *torsion*. They are described by a centerline λ going from *sphere 0* to *sphere 1*; two perpendicular unitary vectors $\vec{u}_i$ and $\vec{v}_i$ are attached to both

spheres and lie in plane S_r; planes S_i are perpendicular to centerline λ and values α_i and β_i express the angles between centerline λ and vectors u_i and v_i. The different degrees of freedom of a link are described here bellow:

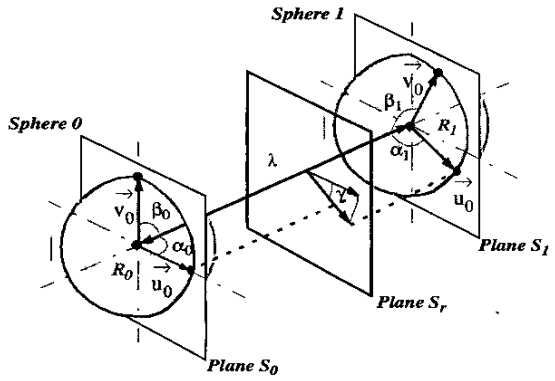

Figure 3 - Two spheres connected with an an elastic link.

Elongation is controlled by an axial spring connecting the centers of each sphere along centerline λ. Compression and elongation forces are applied on both spheres. See figure 4.

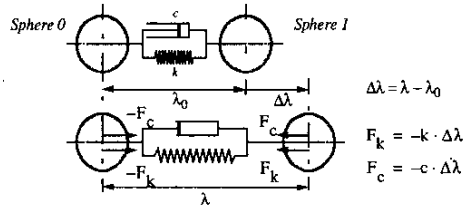

Figure 4 - Axial damped spring controlling elongation.

Flexion is expressed for each sphere by angles α_i and β_i around axis $\vec{v}_i$ and $\vec{u}_i$. Four angular springs control the orientation of each sphere in relation with the centerline λ. Forces and torques applied on each sphere are illustrates in figure 5.

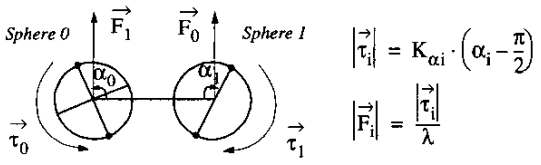

Figure 5 - Illustration of flexion arround axis $\vec{u}$

Finally, torsion is managed by an individual angular spring around centerline λ. Torsion angle γ is measured by projecting vectors $\vec{u}_0$ and $\vec{u}_1$ on plane *Sr* as illustrated in figure 3. Torques issued from torsion are applied on center of both spheres.

2.3 Surface Modeling

The surface model is created in two stages. First, point masses are set on the mesh nodes (vertices) and damped springs on the mesh edges, Secondly, all the nodes are connected to the skeleton by searching for the nearest link or nearest sphere. Two types of connections may occur:

- If a sphere center happens to be the nearest skeleton point, the surface node is directly attached via an elastic link to the reference frame *R* of the corresponding sphere. When the sphere rotates or translates, the surface mesh automatically follows the motion.
- If the nearest skeleton point happens to be located on a link, the surface node is projected and attached onto the centerline λ. During the deformation of a link, length λ is modified due to elongation or compression; nodes connected along the link are simply redistributed along the link. During torsion, the surface nodes rotate around the centerline depending of their position along λ a torsion angle γ.

Any forces applied on the surface triangles first generate a local deformation; afterwards, thanks to surface-skeleton links, forces propagate to the global model and affect the overall shape of the object.

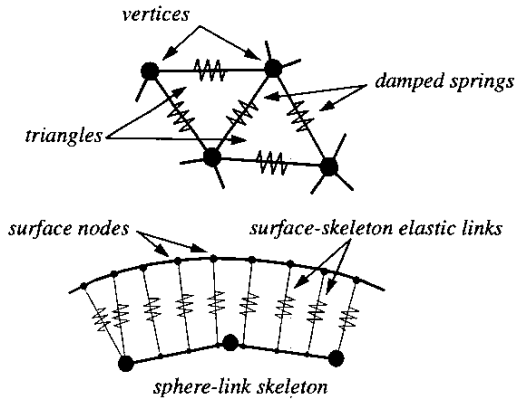

Figure 6 - The surface model is formed by a network of springs located at each triangle edge of the object. The surface nodes are connected to the skeleton via elastic links.

3 Multiple Body Collision

Fast and accurate collision detection between general polygonal models is a fundamental problem in computer-simulated environments. Most of the previous work in collision detection between polygonal models has focused on algorithms for convex polyhedra. However, they are not

applicable in the case of deformable objects, since these structures are generally non-convex, and deform over time. Among the collision detection methods that are applicable to more general polygonal models, almost all of the optimizations rely on a pre-computed hierarchy of bounding volumes. The solutions range from axis-aligned box trees, sphere trees, to BSP trees. All these techniques, which perform very efficient rejection tests, may considerably slow down when objects are very close, causing interactive applications to become unresponsive and thus ineffective.

In our framework, we simplify collision detection between deformable objects by only considering impacts between filling spheres. This method requires much less computation compared to other techniques searching for collisions between triangles. Elastic collisions are computed between colliding spheres and directly influence the volumetric model. Since no collision detection is performed directly on the object's mesh, inaccuracies may be observed depending of the resolution of the skeleton. Figure 7 illustrates two objects in contact with each other; the contour of *object 1* slightly interpenetrates *object 2*. Even if inaccuracies are noticeable, this method does not degrade the overall performance of the simulation. This method will not perform correctly for skeletons that do not enclose the overall volume of the object (See Dolphin in figure 10); in these cases, objects may intersect via the empty gaps contained between adjacent spheres.

Complex scenes, containing several dozens of objects, may require computing hundreds or even thousands of collision checks between spheres, thus affecting the overall performance of the simulation. To reduce computational time during collision checking, we can consider for certain applications (i.e. a surgery simulator), that some objects or bodies only move in a limited range of space. Given this condition we can draw, for each object, a limited list of neighbors with whom potential collisions may occur and is worth checking.

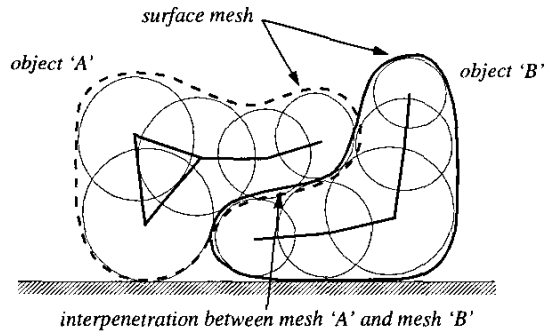

Figure 7 - Collision detection between two objects using the spherical representation.

4 Haptic Interaction

Early haptic rendering systems modeled surface contacts by generating a repulsive force proportional to the amount of penetration into an obstacle. While these penalty-based methods work well to model simple obstacles, such as planes or spheres, a number of difficulties are encountered when trying to extend these models to display more complex environments.

An alternative is not to look at the penetration of the user's finger into the object at all, but instead to constrain the motions of a substitute virtual object. In the method proposed by Ruspini et. Al [4], a representative object, a "proxy," substitutes in the virtual environment for the physical finger or probe. The "virtual proxy" can be viewed as if connected to the user's real finger by a stiff spring. As the user moves his/her finger in the workspace of the haptic device he/she may pass into or through one or more of the virtual obstacles. The proxy, however, is stopped by the obstacles and quickly moves to a position that minimizes its distance to the user's finger position. The haptic device is used to generate the forces of the virtual spring which appears to the user as the constraint forces caused by contact with a real environment. This approach is similar to the method for the "gob-object" first proposed by Zilles et. al [3] but does not require apriori knowledge of the surface topology. In our framework, the forces resulting from the finger-proxy model are directly applied to the mass-points composing triangles in contact with the finger. A weighted distribution based on the contact position of the finger on the triangle determines the amount of force applied to each vertex. An example of the finger-proxy interacting with a deformable object is illustrated in figure 8.

Because virtual objects are normally constructed with a large number of triangles, a naïve test based on checking if each primitive is in the path of the proxy would be prohibitively expensive. Instead a hierarchical bounding representation for the object is constructed to take advantages of the spatial coherence inherent in the environment. In our framework, we make use of the filling spheres model to generate a bounding box representation. Each sphere is attributed a second larger radius to contains all linked triangles. During collision check, we first search for any intersecting spheres. If one is selected we then check for eventual collision with every triangle inside the sphere. By directly using our spherical model for collision detection, we remove any extra computation necessary for updating a bounding box representation when an object changes shape.

A cache is maintained to avoid calling the low-level check multiple times for the same primitive during the same iteration. Some spatial coherence, such as a list of sphere or triangle neighbors, can be used to reduce the

computation time between successive calls to the collision algorithm.

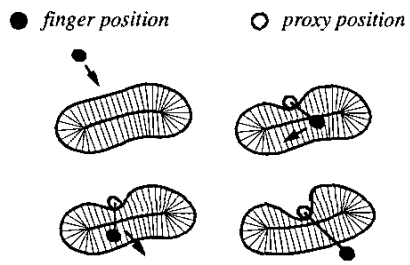

Figure 8 - finger-proxy example on a deformable object

5 Implementation

Our current system runs on a Dual 1-GHZ Pentium III Computer under Windows XP. Two main threads compose the backbone of our application: the *haptics engine* and the *dynamics simulator*. The separation of both processes was first proposed by Adachi et al. [13]. Decoupling the low-level force servo loop from the high-level control is important since the haptic servo loop must run at a very high rate, typically greater than 1000 Hz, to achieve a high quality force display. Our dynamics simulator typically runs at a much slower rate (~20-100Hz). The haptics engine uses the finger/proxy model, presented in paragraph 4, to compute forces between user's fingers and virtual objects; this information is transferred to the dynamics engine to generate physical interactions. Since the physical models are only updated at 20-50 Hz, the haptics engine performs a blending transition between the two last computed frames to avoid user to feel discrete steps between two iterations of the dynamics simulator . Even though a small delay is introduced (< 50 ms) between the user input and the simulation output, the effect is not perceived if time-steps remain small enough (< 100ms).

All physical models are generated offline for each object and loaded into memory during the startup process of the application. An XML script file describes the position, orientation and physical characteristics of each object in the scene.

6 Results

We developed several applications to demonstrate each component of the framework, some of which are available on our website.

In figure 10, we present a virtual Dolphin with its corresponding skeleton generated by placing filling spheres along the medial axis transform. Links interconnect neighbor spheres together. The mesh is composed of 1120 triangles and a connection between the mesh and the skeleton is performed for every vertex. A dynamic behavior was programmed for each Dolphin by controlling the leading sphere, located in the head of the Dolphin, along a sine wave function. The effects were propagated along the rest of the body and gave the illusion of a Dolphin swimming in an ocean. To demonstrate the low amount of computation required for this application, we performed a real-time simulation on a Pentium Pro 200 MHz computer equipped with an AccelGraphics video card for fast OpenGL rendering. A second application was performed on a much faster computer (Dual 1-GHz Pentium III), with a complete medical landscape containing a liver, digestive system, stomach and vesicle. 411 spheres composed the dynamic skeletons of the organs. Surface and global deformation were performed in real time, as well as haptic interaction between the user and the virtual environment. Figure 11 presents a close view of the user grabbing the surface of a gallbladder. We observe the local and global deformations being rendered.

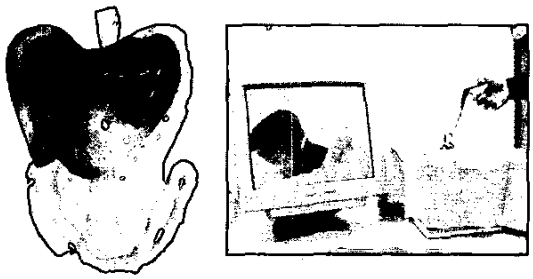

Figure 9 - Laparoscopic surgery simulator. (Left) Overall view of the virtual organs. (Right) Workstation and force feedback haptic device

7 Conclusion

In this paper, we presented a technique to compute deformations of virtual objects by decoupling surface and volumetric representation. By introducing a new method based on six degrees of freedom filling spheres, we generated realistic global deformations on complex models with minimal computation. We also presented a methodology, based on the medial axis transform, to construct physical models at adjustable resolutions. Finally, we combined the global model with a surface spring model to handle local deformations.

While this approach has shown promising results for applications requiring real-time interactivity, nevertheless many problems remain when cutting procedures are realized. In these situations, updating the mesh and skeleton is a complex task and often computationally expensive.

Future work includes an optimized mechanism to update the model during cutting-procedures, volume conservation and comparison of this approach with finer methods such as FEMs.

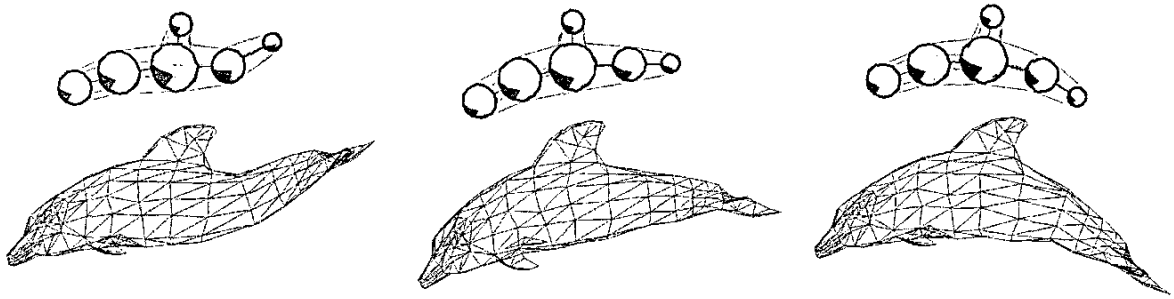

Figure 10 - Representation of a dolphin with a skeleton composed of 6 spheres and 5 links

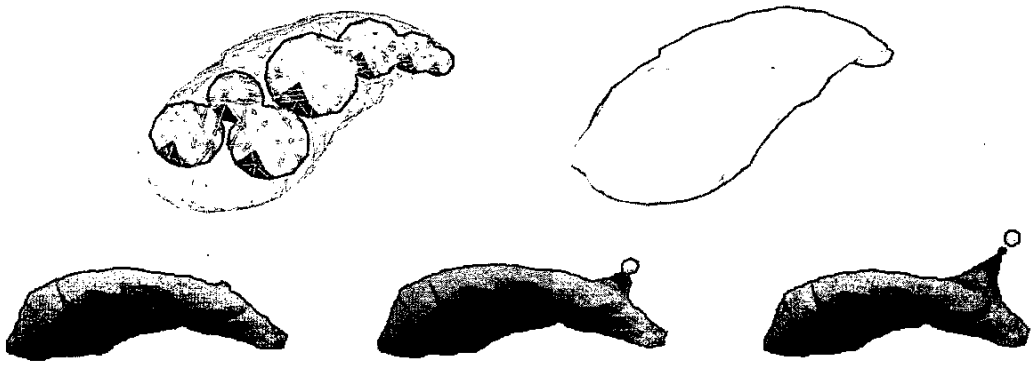

Figure 11 - Surface deformation of a virtual vesicule composed of 466 triangles. The skeleton is composed of 7 spheres and 7 links

References

[1] Francois Conti, *"Deformation of virtual objects"*, Master thesis, EPFL, Lausanne, Switzerland, 1999.

[2] 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.

[3] C.Zilles, J.Salisbury. *"Constraint-based God-object Method for Haptic Display"*, ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems. Dynamic Systems and Control, 1994, vol.1., pp 146,150.

[4] Ruspini, Diego, Krasimir Kolarov, Oussama Khatib, "The Haptic Display of Complex Graphical Environments." SIGGRAPH 97 Proceedings, (August 1997), pp. 345-352.

[5] Nina Amenta, Sunghee Choi and Ravi Kolluri. *"The power Crust"*. ACM Symposium on Solid Modeling and Applications, 2001

[6] Hoppe H. *"Progressive Mesh"*, Proceeding of ACM SIGGRAPH, pp. 99-108, 1996

[7] Xuaeto Li, Tong Wing Woon, Tiow Seng Tan and Zhiyong Huang *"Decomposing Polygon Meshes for Interactive Applications"*

[8] David Bourguignon and Marie-Paul Cani. *"Controlling Anistropy in Mass-Spring Systems"*

[9] Sarah F. F. Gibson and Brin Mirtich. *"A Survey of Deformable Objects in Computer Graphics"*

[10] A. Van Gelder. *"Approximate Simulation of Elastic Me branes by Triangulated Spring Meshes."*. Journal of Graphics Tools, 3(2):21-41, 1998

[11]Kolja Kaehler, Joerg Haber and Hans-Peter Seidel. *"Dynamic Refinement of Deformable Triangle Meshes for Rendering"*

[12] Jaso J. Corso, Jatin Chhugani and Allison M Okamura. *"Interactive Haptic Rendering of Deformable Surfaces Based on the Medial Axis Transform"*. Eurohaptics, UK, July 8-10, 2002 Eurohaptics Proceedings, pp. 92-98

[13] Adachi, Y., Kumano, T., Ogino K., *"Intermediate Representation for Stiff Virtual Objects."* Proc. IEEE Virtual Reality