An Implicit-Based Haptic Rendering Technique

Laehyun Kim   Anna Kyrikou   Gaurav S. Sukhatme   Mathieu Desbrun

University of Southern California,
{*laehyunk, kyrikou, gaurav, desbrun*}*@usc.edu*

Abstract

We present a novel haptic rendering technique. Building on previous work, we propose a haptic model based on a volumetric description of the geometry of an object. Unlike previous volumetric approaches, we also find a virtual contact point on the surface in order to derive a penalty force that is consistent with the real geometry of the object, without introducing force discontinuity. We also demonstrate that other surface properties such as friction and texture can be added elegantly. The resulting technique is fast (a constant 1000 Hz refresh rate) and can handle large geometry models on low-end computers.

1 Introduction

A haptic display device is a force output device as well as a point input device in 3-D space. It renders the virtual object tangible and provides an intuitive interface with a virtual environment. When the user touches a virtual object, the haptic rendering algorithm generates an adequate force field to simulate the presence of the object and surface properties such as friction and texture.

The haptic rendering methods can be classified into roughly two groups according to the surface representation they use: geometric haptic algorithms [23, 19, 13, 20] used to render surface data and volume haptic algorithms [1, 9, 11] used for volumetric data.

Our novel haptic rendering algorithm takes advantage of both the geometric (B-rep) and the implicit (V-rep) surface representations for a given 3D object. The geometric model can effectively represent the interface between the object in 3D and the rest of the scene, while an implicit surface representation has many properties which can highly benefit to the haptic rendering algorithm. The novelty of our technique therefore lies in exploiting both representations to derive a fast and accurate haptic rendering technique.

We will proceed with a short discussion on previous related work in section 2. Some background information about implicit surfaces will be presented in section 3. In section 4, we will describe the force field generation and surface properties such as friction and haptic texture. In section 5 we will present applications for both geometric and implicit representation based model. We conclude with some results and future perspectives of this algorithm in section 6.

2 Related Work

Traditional haptic rendering methods are based on geometric surface representations which mainly consist of triangle meshes. One interesting approach for the geometric models is the penalty-based method [13, 14] which suffers from a strong force discontinuity and push-through for thin and small objects [23, 19]. In order to overcome these limitations, Zilles and Salisbury [23] introduced a constraint-based “god-object” method in which the movement of a god-object (a virtual contact point on the surface) is constrained to stay on the object’s surface. Ruspini *et al.* [19] proposed another constraint-based and simulation method for additional surface properties such as friction and haptic texture. They used an implementation of Force Shading [18] in which a surface normal is obtained by interpolating the normals from the vertices of the mesh.

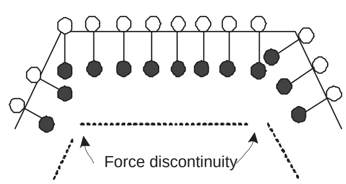
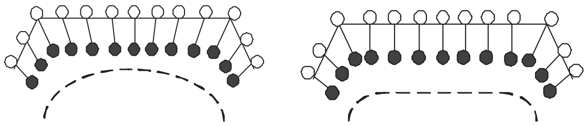


Figure 1: *Force discontinuity in constraint-based approach*

The constraint-based approach, however, still suffers from force discontinuity (see Figure 1) and does not scale well with the complexity of the object’s surface. Force Shading solves the force discontinuity problem, but introduces a feeling of rounded surfaces due to the discrepancy between the haptic force field and the actual normal field of the surface [8, 19] as sketched in Figure 2a.

Note that haptic rendering algorithms for geometric models are not applicable to volumetric data without a prior conversion, using a Marching Cubes algorithm [12] for instance. However, the resultant geometric models usu-



(a) (b)

Figure 2: *Comparison of two haptic methods without force discontinuity (a) Force shading method (b) Our approach*

ally contain a large number of polygons, which makes it impractical.

In the volume haptic rendering, instead, the force field is computed directly from the volume data. A haptic rendering algorithm for volumetric data was first introduced by Iwata and Noma [9]. They used the gradient of the potential value to calculate the force. However, in this volume haptic rendering technique and those that followed [1, 11], the amount of force was linearly proportional to the potential value. This approximation does not take into account the distance to a virtual contact point on the surface - or in other words, it is a good approximation only when the haptic device is extremely close to the real surface. As a result, the haptic surface does not match the visible surface, as sketched in Figure 3a.

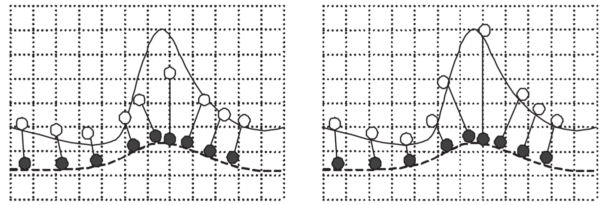
2.1 Contributions

We employ an implicit surface representation to haptically render a geometric model to take advantage of the implicit representation as well as the geometric representation. In our algorithm, the user ”sees” a geometric model and ”feels” an implicit surface which wraps around the geometric model (Figure 4).

Our approach avoids the force discontinuity around volumetric boundaries (edges and vertices) in a geometric model without a feeling of rounded surfaces (Figure 2b) using an interpolation function in a volumetric model, which was first introduced by Avila [1].

Another contribution is that we use a simple variation of Avila’s volume haptic method [1] to obtain the correct force magnitude. In previous volume haptic algorithms [1, 11], the force magnitude is a function of the potential value. This approximation may not allow the user to feel stiff objects (Figure 3a). In order to solve the problem, we find a virtual contact point on the surface to render the surface more accurately and robustly (Figure 3b).

Haptic texturing, in our algorithm, is implemented by mapping a texture pattern directly into the implicit representation unlike previous haptic texturing methods [2, 13, 20] which modulate the friction or perturb the surface normal. As a result, the geometry of the implicit surface is changed and it can express texture geometry explicitly



(a) (b)

Figure 3: *computing the force magnitude (a) approximation in previous methods (b) our approach*

without additional computations.

Furthermore, our algorithm is fast and stable, and can be used for huge models on low-end computers.

3 Implicit surface representation

3.1 Definition

The implicit representation of the external surface S of an object is described by the following implicit equation [3]

$$S = \{(x, y, z) \in R^3 | f(x, y, z) = 0\}$$

where f is the implicit function (also called potential), and (x, y, z) is the coordinate of a point in 3D space.

If the potential value is 0, then the point (x, y, z) is on the surface. The set of points for which the potential value is 0 defines the implicit surface. If the potential is positive, then the point (x, y, z) is outside the object (red points in Figure 4a). If $f(x, y, z) < 0$, then the point (x, y, z) is inside the surface (green points in Figure 4a).

In our algorithm, we define the volumetric representation using a discrete potential stored on a 3D regular grid. The potential value of each point in a grid is a signed scalar value which indicates the proximity to the surface. Potential values inside a cube of the grid is computed using a trilinear interpolation between the eight values at the corner of the cube. Now, the inside/outside property of the potential function makes the collision detection between the tool tip of haptic display device and the implicit surface trivial, since we know (at fixed computational cost) the sign of the potential.

3.2 Surface normal

The surface normals of an implicit surface can be obtained using the gradient of the implicit function as follows [3]:

$$n = \nabla f / \|\nabla f\| \tag{1}$$

$$\nabla f = [\frac{df}{dx}, \frac{df}{dy}, \frac{df}{dz}] \tag{2}$$

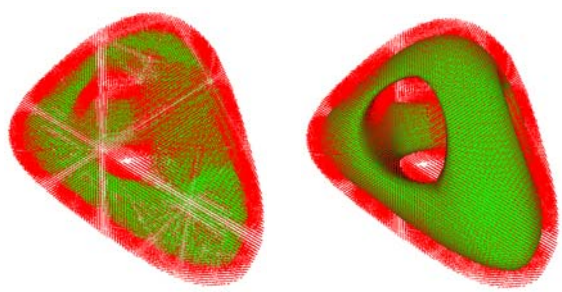
The surface normal of a given point inside the surface can be computed by interpolating the gradients of the 8 neighboring points around the point. This property is crucial to achieve smooth changes of the force direction in our algorithm.

3.3 Closest point transform

The potential value of each point is pre-computed using the closest point transform (CPT) [15]. CPT converts an explicit representation of a geometric surface into an implicit one.

A fast algorithm for computing the closest distance was proposed by Mauch [15]. The algorithm computes the closest point to a surface and its distance from it by solving the Eikonal equation using the method of characteristics. The computed distance is accurate and the complexity is linear in the number of grid cells and surface complexity.

In our algorithm, Mauch's CPT algorithm is used to generate the potential value of each point. The user can select the resolution of the grid depending on the surface complexity of objects (see Figure 4), or depending on the computational power available.



(a) (b)

Figure 4: *Closest point transform (a) potential values without a geometric model in a grid (b) wrapping around the model*

4 Haptic Rendering Model

In this section we give a more detailed presentation of our haptic model including collision detection, force generation, and surface properties like friction and haptic texture.

4.1 Collision detection

In geometric haptic rendering models, collision detection is not trivial to compute. One of the most popular collision detection algorithms in geometric haptic rendering is H-Collide [7]. It uses a hybrid hierarchy of spatial subdivision and OBB trees. Ruspini et al. [19] used a bounding sphere hierarchy to detect collisions.

Using the implicit representation, collision detection becomes trivial due to the inside/outside property. We can obtain the proximity to the surface by interpolating the potential values of the 8 neighbor points around the tool tip. If the distance becomes 0 or changes sign, a collision is detected. The complexity is constant since we are using a regular grid and is independent of the resolution of the grid.

4.2 Friction-less Model

Force Direction. As mentioned before, penalty-based approaches [13, 14] have many limitations such as finding the nearest surface, strong force discontinuity, and push-through of thin objects. Constraint-based approaches [23, 19] overcome these problems to some extent. These approaches, however, still suffer from force discontinuity (see Figure 1). The force discontinuity generally occurs when the direction and/or amount of the force are changed suddenly around volumetric boundaries such as edges on the surface. The force discontinuity is a crucial problem in a haptic rendering algorithm since the human sense of touch is sensitive enough to notice even small force discontinuities.

In the implicit surface representation, we can obtain smooth surface normals as the tool tip moves along the surface. When the tool tip is inside the surface, the position of the tool tip lies on a certain isosurface inside the real surface. The isosurface works like an inner constraint of the surface. When the user touches the surface of an object, the algorithm first computes the gradient of each point of 8 neighbors around the tool tip in a 3D grid. Then the gradient at the position of the tool tip is computed by interpolating the neighbor's gradients. The resulting gradient is equal to the surface normal of "a" virtual contact point on the surface and becomes the direction of the force (see Figure 2b). The next section shows how to find the exact virtual point in the direction of the force, so that the force magnitude is consistent with the surface we have at hand.

Force Magnitude. In the previously introduced volume haptic algorithms [1, 9, 11], the force magnitude has been approximated using the potential value. However, the potential value may not be proportional to the virtual contact point on the surface. This occurs for instance in thin convex and concave surface of rugged objects (see Figure 3a). As a result, the user usually feels the surface smoother than he sees it.

In our algorithm we first find the virtual contact point on a surface in order to determine the amount of the force. This means that the force magnitude is not a function of the arbitrary potential value. The virtual contact point is constrained by, and moves along the surface, just as the constraint-based approach for the geometric representation (see Figure 3b). The contact point is found as the intersection point between the surface and a ray along the computed force direction. The position of that point can be quickly calculated by binary search. As the tool tip is usually very close to the surface, the computation required is extremely simple (usually only a few steps along the ray suffice).

Once the virtual contact point is found, a spring-damper model [22] is used in order to compute the force vector that tries to keep the virtual contact point and the tool tip at the same position:

$$\vec{F} = (\mathbf{p_c} - \mathbf{p_t}) * k - \vec{V} * b \tag{3}$$

where $\vec{F}$ is the force vector, $\mathbf{p_c}$ is the coordinate of the contact point, $\mathbf{p_t}$ is the coordinate of the tool tip, k is stiffness, $\vec{V}$ is velocity of the tool tip, and b is viscosity (see Figure 5). Spring stiffness has a reasonably high value and viscosity is to prevent oscillations. By modulating stiffness and viscosity, the user can vary surface stiffness.

To provide more robustness, we also threshold the resulting magnitude. When the tool tip penetrates deeply inside the surface with a low stiffness, instability can occur in a very complex surface since isosurfaces are getting smoother as the depth of penetration increases.

4.3 Adding Friction to the Model

If the model has no friction (viscosity), it creates the feeling of a very slippery surface, since the direction of the force vector is always perpendicular to the surface. Therefore, the algorithm should incorporate a friction term in order to simulate various surfaces with different friction properties.

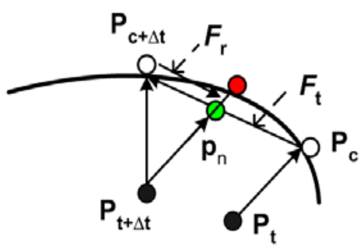


Figure 5: *The new virtual contact point due to friction.*

In our algorithm, friction is implemented by limiting the movement of the virtual contact point like in the constraint-based method. The friction term takes into account a friction coefficient and the depth of penetration.

$$f_v = f_c * (1 + (\|\mathbf{p_{c+\triangle t}} - \mathbf{p_{t+\triangle t}}\|) * d) \tag{4}$$

where f_v is the friction term which ranges from 0.0 to 1.0, f_c is the friction coefficient, $\|\mathbf{p_{c+\triangle t}} - \mathbf{p_{t+\triangle t}}\|$ is the penetration depth and d is a depth constant. Due to the depth term in the equation above, the user feels a stronger retarding force as he/she is moving inside the object.

By using the friction term f_v, we can compute the retarding force $\vec{F_r}$ and the new contact point as follows:

$$\vec{F_t} = \mathbf{p_{c+\triangle t}} - \mathbf{p_c} \tag{5}$$

$$\vec{F_r} = -\vec{F_t} * f_v \tag{6}$$

$$\mathbf{p_n} = \mathbf{p_c} + (\vec{F_t} + \vec{F_v}) \tag{7}$$

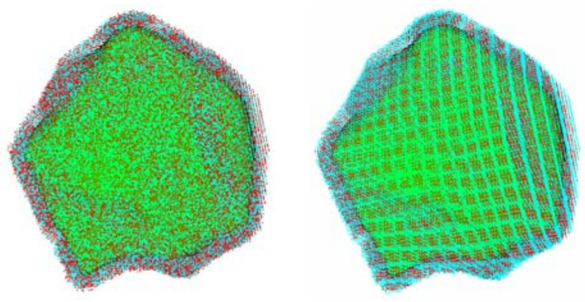
where $\vec{F_t}$ is the tangential force, $\mathbf{p_{c+\triangle t}}$ is the current contact point coordinate, $\mathbf{p_c}$ is the previous contact point coordinate, $\vec{F_r}$ is the retarding force and $\mathbf{p_n}$ is the new position of the tangential force after applying friction (the green point in Figure 5). The new position $\mathbf{p_n}$, however, may not lie on the surface. We have to find the new contact point (the red point in Figure 5) which is on the surface and intersects with a ray along the new surface normal vector $(\mathbf{p_n} - \mathbf{p_{t+\triangle t}})$. The final force is finally calculated using the equation 3.

4.4 Haptic texture

Haptic texturing is the term used to describe the way we can simulate surface roughness. It can enrich the user interaction with a haptic device just as graphical texture enhances visual realism.

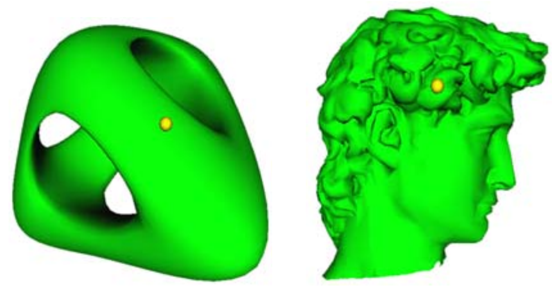
In previously introduced algorithms, haptic textures are implemented by modulating the surface friction and/or local surface normals. Minsky et al. [17] first demonstrated simulation of some haptic textures by lateral force fields proportional to the local gradient of the textured surface. Basdogan et al. [2] implemented a bumpy surface by perturbing the direction and magnitude of the surface normal. Several approaches are based on the stick-slip friction model [13, 20]. In this model, the tool tip of the haptic device is 'stuck' (restrained) by the means of a static friction until the user applies enough force to overcome this static friction. Then, the tool tip moves away from the sticking point and 'slips' until it meets the next snagging point.

In our algorithm, haptic texturing is simulated by applying Gaussian noise (Figure 6a) and texture patterns (Figure 6b) *directly to the potential value of each point in the 3D grid*, without any need for preprocessing. No modification to the existing algorithm is necessary in order to accommodate the new texture features of the surface. Moreover, there is no additional computational cost imposed due to haptic texturing since the direction and amount of the force are computed dynamically as the tool tip moves along the surface, whether or not this one has been modified by additional textures.



(a) (b)

Figure 6: Haptic Texture (a) Gaussian noise (b) Lattice pattern



(a) (b)

Figure 7: Haptic display for geometric models (a) model with non-zero genus (3328 triangles) (b)David model, with complex geometry (11820 triangles).

5 Results

As mentioned before, our algorithm is applicable to both geometric models and volumetric data without any modifications. This section presents two applications. One application is the direct haptic simulation on geometric models, and the other is an example of how this algorithm is used in a rapid modeling prototyping toolkit (VST), based on implicit representation.

In our algorithm, the collision detection and the force generation are integrated into the servo-loop which sends the force directly to the PHANTOM haptic device at 1000 Hz. A sample program is available at "http://www-grail.usc.edu/Haptic".

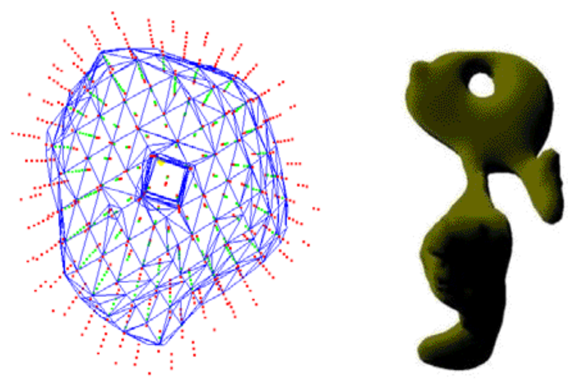
5.1 Haptic simulation for geometric models

In the geometric representation, the user "sees" a geometric model and "feels" the implicit surface which is converted from the geometric model by the fast CPT (see Figure 4). The preprocessing stage to compute the distance field is generally a matter of seconds (2-4 seconds on a 150x150x150 grid for two models in Figure 7), and needs to be done only once before the haptic simulation. Due to the use of only local computations, we can adjust the resolution of a grid without any noticeable performance degradation on low-end computers.

5.2 Virtual sculpting based on the implicit representation

We developed a virtual sculpting toolkit (VST) as an alternative to existing digital sculpting implementations. Our haptic rendering algorithm is integrated into VST to haptically render the implicit surface and to intuitively manipulate the deformation (Figure 8).

VST is based on the work by Desbrun and Cani-Gascuel [4] in which they first introduced the concept of using active implicit surfaces in order to animate objects defined using an implicit potential on a regular grid. We use active implicit surfaces to compute the deformations created by the haptic device. When we apply the tool on a region close to the actual surface and for the period of time the tool is activated, it creates a *force field* which propagates through time to the neighboring nodes like a wind field, altering their potential values. Using VST, we created the snoopy model (Figure 8b) which has 7468 triangles on a 70x70x70 grid. All surface modification is achieved in real time on a 1GHz Pentium III CPU and a Geforce 3 64 mb video card.



(a) (b)

Figure 8: Virtual sculpting tool (a) making a hole in a model (b) snoopy model created by VST

6 Conclusion and future works

Combining geometric and volumetric haptic rendering methods takes advantage from both approaches. Our haptic algorithm is mainly based on an implicit surface representation which represents the surface with potential values in a 3D regular grid. Thanks to the implicit properties, the user can feel a smooth surface without force discontinuity. In addition, we compute the position of a virtual contact point, which is constrained to be on the surface just like in constraint-based approaches. As a result, the user feels the real geometry of the surface unlike previous volumetric haptic simulations.

Our algorithm also simulates effectively surface properties like friction, stiffness, and haptic texture. Especially, haptic texture is implemented by directly modulating the potential values of the grid.

The algorithm allows to update forces at the 1 kHz rate due to the fast computation of the collision detection and force model.

The current method is based on an implicit representation in a regular spacing grid where the potential is a scalar. We would like to test our method on the refined implicit implementation proposed by Kobbelt *et al.* [10] where models produced are significantly more detailed and also see how it performs if we use Adaptively Sampled Distanced Fields (ADF) proposed by Frisken et al [5].

In addition, we are going to extend our haptic algorithm to support image-based haptic texturing [2] which allows the user to feel a texture image and haptic painting [6] which draws directly on a 3D model using a haptic device.

References

[1] R. S. Avila, L. M. Sobierajski, “A Haptic Interaction Method for Volume Visualization” IEEE Visualization’96, Oct. 1996.

[2] C. Basdogan, C. Ho, M. A. Sriniviasan, “A Ray-based Haptic Rendering Technique for Displaying Shape and Texture of 3D Objects in Virtual Environments”, Proceedings of the ASME Dynamic Systems and Control Division. 1997

[3] J. Bloomenthal, et al. “Introduction to Implicit surface”, Morgan Kaufmann Publishers, Inc. 1997

[4] Mathieu Desbrun, Marie-Paule Cani-Gascuel, ”Active Implicit Surface for Animation” In Graphics Interface’98

[5] S. Frisken, R. Perry, A. Rockwood, T. Jones, “Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics”, In SIGGRAPH’00, pp. 249-254, 2000

[6] A. Gregory, S. Ehmann, and M. C. Lin. “inTouch: Interactive multiresolution modeling and 3D painting with a haptic interface”. Proceedings of IEEE VR Conference, 2000.

[7] A. Gregory, M. Lin, et al. “H-Collide: A Framework for Fast and Accurate Collision Detection for Haptic Interaction”, In IEEE Virtual Reality, 1999

[8] J. M. Hollerbach and D. E. Johnson, Virtual Environment rendering, to appear in Human and Machine Haptics, M. Cutkosky, R. Howe, K. Salisbury, and M. Srinivasan, eds., MIT Press, 2000.

[9] H. Iwata and H. Noma, “Volume haptization” In Virtual Reality, pp. 16-13. IEEE ’ 93 Symposium on Research Frontiers in, 1993

[10] L. P. Kobbelt, M. Botsch, U. Schwanecke, H. Seidel, “Feature-Sensitive Surface Extraction From Volume Data” In SIGGRAPH’01, pp. 57-66, 2001

[11] D. A. Lawrence, C. D. Lee, L. Y. Pao, “Shock and Votex visualization Using a Combined Visual/Haptic Interface”, IEEE Visualization’00 2000

[12] W. E. Lorensen and H. E. Cline, “Marching Cubes: a high resolution 3D surface construction algorithm”, Computer Graphics, 21 (4), 163-9;1987

[13] W. Mark, S. Randolph, M. Finch, J. V. Verth, R. M. Taylor II. “Adding Force Feedbackto Graphics Systems: Issues and Solutions” In SIGGRAPH’96, pp. 447-452, August 1996

[14] Thomas H. Massie, J. K. Salisbury, “The Phantom Haptic Interface: A Device for Probing Virtual Objects” ASME Haptic Interfaces for Virtual Environment and Teleoperator System Nov. 1994.

[15] Sean. Mauch, “A Fast Alogorithm for Computing the Closest Point and Distance Transform” http://www.cco.caltech.edu/ sean/closestpoint/ closept.html.

[16] M. L. McLaughlin, J. P. Hespanha, G. S. Sukhatme, “Touch In Virtual Environments”, Prentice Hall PTR, 2001

[17] M. D. R. Minsky, “Computational Haptics: The Sandpaper System for Synthesizing Texture for a Force-Feedback Display”, PhD thesis, MIT, June 1995

[18] H. B. Morgenbesser, ”Force Shading for Haptic Shape Perception in Haptic Virtual Environment” M. Eng. thesis, MIT, September 1995.

[19] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. “The haptic display of complex graphical environment” In SIGGRAPH 97 conference proceedings, volumn 1, pp. 295-301, August 1997

[20] K. Salisbury, et al., “Haptic rendering: programming touch interaction with virtual objects” Symposium on Interactive 3D Graphics, pp. 123-130, 1995

[21] Juhani Siira, Dinesh K. Pai, “Haptic Texturing - A Stochastic Approach”, IEEE International Conference on Robotics and Automation, 557-562, Minnesota, 1996.

[22] A. Yoshitaka, T. Kumano, K. Ogino, “Intermediate Representation for Stiff Virtual Objects”, IEEE VRAIS’95 pp. 203-210 March 1995

[23] C. Zilles, J.K. Salisbury, “A Constraint-based God-object Method For Haptic Display”, ASME Haptic Interfaces for Virtual Environment and Teleoperator System 1994.