DSC-Vol. 61, Proceedings of the ASME
Dynamic Systems and Control Division
ASME 1997

A Ray-Based Haptic Rendering Technique for Displaying Shape and Texture of 3D Objects in Virtual Environments

Cagatay Basdogan
Chih-Hao Ho
Mandayam A. Srinivasan

Laboratory for Human and Machine Haptics
Department of Mechanical Engineering and
Research Laboratory of Electronics
Massachusetts Institute of Technology
Cambridge, MA, 02139
basdogan, chihhao, srini @mit.edu
http://touchlab.mit.edu

Abstract
In this paper, we propose a new *ray-based* haptic rendering method for displaying 3D objects in virtual environments. We have developed a set of software algorithms that work with a force reflecting haptic interface and enable the user to touch and feel arbitrary 3D polyhedral virtual objects. Using the interface device and the suggested model, the user feels as if exploring the shape and surface details of objects such as textures. The components of the model include a hierarchical database for storing geometrical and material properties of objects, collision detection algorithms, a simple mechanistic model for computing the forces of interaction between the 3D virtual objects and the force-reflecting device, and a haptic filtering technique for simulating surface details of objects such as smoothness and texture. The developed algorithms together with a haptic interface device have several applications in areas such as medicine, education, computer animation, teleoperation, entertainment, and rehabilitation.

1. Introduction
Humans explore objects in many steps. Typically, we first visually scan the environment with our eyes to locate the position of the object. We then touch the object to feel its general shape. Finally, a more careful manual exploration is made to investigate the surface and material characteristics of the object. Manual sensing of shape, softness, and texture of an object occurs through tactile and kinesthetic sensory systems that respond to spatial and temporal distribution of forces on the hand. Recent advances in virtual reality and robotics enable the human tactual system to be stimulated in a controlled manner through force feedback devices, also known as haptic interfaces. A haptic interface is a device that enables manual interaction with virtual environments or teleoparated remote systems. They are employed for tasks that are usually performed using hands in the real world (Srinivasan, 1995). Force-reflecting haptic devices generate computer-controlled forces to convey to the user a sense of natural feel of the virtual environment and objects within it. In this regard, haptic rendering can be defined as the process of displaying computer controlled forces on the user to make him or her sense the tactual feel of virtual objects (Salisbury et al., 1995). Haptic rendering is a part of an emerging field we call *computer haptics* (analogous to computer graphics), that is concerned with the techniques and processes associated with generating and displaying haptic stimuli to the human user (Srinivasan and Basdogan, 1997).

This paper describes a general method for haptic rendering of 3D objects in virtual environments. For the sake of simplicity, we restrict ourselves to static environments composed of rigid objects. When these virtual objects are explored with a generic stylus of the force feedback device, the user can feel their shape and texture. The proposed method has four major components:

- a hierarchical database for storing the geometrical and material properties of virtual objects
- a set of algorithms for detecting the collisions between the simulated probe of the haptic device and the 3D virtual objects
- a mechanistic model for estimating the reaction forces that arise from interactions between the simulated probe of the haptic device and the virtual object
- a filtering technique for modifying the direction and magnitude of the reaction forces to provide the user with the tactual sense of surface details such as smoothness and texture.

The developed techniques are suitable for simulating haptic interactions between a tool and 3D objects in synthetic environments. The graphical models of 3D objects and the generic probe (stylus) of the haptic device are composed of triangular elements and displayed on the computer screen. The hardware components of our set-up include an IBM compatible personal computer, running Open Inventor (Template Graphics Inc.), to display the graphical model of the 3D virtual environment, and a force feedback device, PHANToM (SensAble Technologies Inc), to convey to the user a sense of touch and feel of virtual objects (see Figure 1) in this environment

2. Haptic Rendering

The goal of haptic rendering is to display the shape, surface and material properties of arbitrary 3D objects in real time through a haptic interface Initial haptic rendering methods focused on displaying object primitives (e.g. cube, sphere, cylinder, etc.). Massie and Salisbury (1994) developed the PHANToM haptic interface device and proposed a simple method for rendering objects. Using this rendering model, the user could interact with primitive objects such as cube, cylinder, and sphere in virtual environments through the end point of the haptic device which is defined as the haptic interface point Later, Zilles and Salisbury (1995) developed a more sophisticated, constraint-based method to render generic polygonal meshes. They defined a new point, namely the "god-object", to represent the location of the ideal haptic interface point. Lagrange multipliers are used to compute the new location of the god-object point (constrained to remain on a particular facet of the object) such that the distance between the god-object and the haptic interface point is minimized..

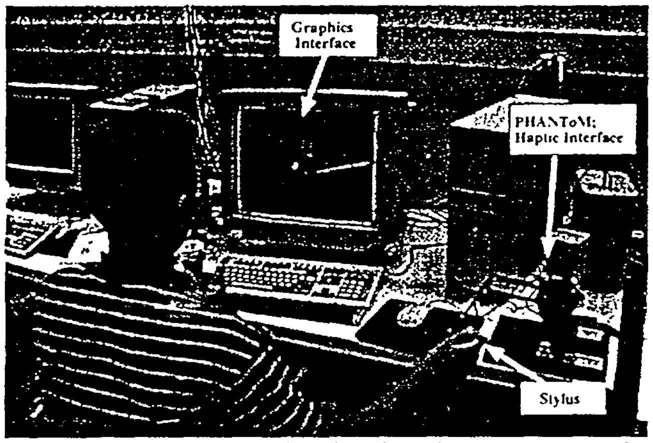


Figure 1. Components of the virtual environment system for multimodal interactions.

We propose a new haptic rendering method that utilizes the ray tracing techniques of computer graphics for detecting collisions between 3D polygonal objects (rigid) and the generic stylus of the haptic device. The stylus is modeled as a line segment whose position and orientation are provided by the encoder signals in the haptic interface. We display the graphical model of the simulated stylus and update its tip and tail coordinates as the user manipulates the actual one (see Figure 1), detect any collisions between the simulated stylus and the virtual object, estimate the reaction force, and finally reflect this force to the user via the haptic device. The proposed method is more suitable for simulating "tool-object" interactions than earlier techniques and takes into account the orientation of the stylus.

2.1 Hierarchical Database and Collision Detection

During the pre-computation phase of our real-time simulations, graphical models of 3D objects, made of triangular elements, are loaded into computer from a data file to construct a database for material and geometrical properties of objects. The information in this database includes the indices of triangular elements, coordinates of vertices of each triangular element and the connectivity of triangular elements in a global coordinate system. Using the information in the database, algorithms compute the bounding box of each object (i.e. a box that is constructed from maximum and minimum global coordinates of the object along the X, Y, and Z axes) and the bounding box and surface normal of each triangular element of the object. As the user manipulates the stylus of the force feedback device, we check for collisions between the simulated stylus and the virtual objects in the environment. In order to speed up the detection of collisions between the simulated stylus and the objects, a simplified mathematical model of the stylus is considered. Although the complete 3D geometrical model of the stylus appears to the user on the computer screen, the stylus is represented as a line segment for the purposes of fast collision computations (see Figure 2a). The tip and tail coordinates of the simulated stylus are updated in the virtual environment as the user manipulates the real stylus. The detection of collisions occur in three consecutive stages:

- collision detection between the simulated stylus and the bounding box of the virtual object
- collision detection between the simulated stylus and the bounding box of the each triangular element
- collision detection between the simulated stylus and the triangular element itself

2.2 Mechanistic Model

The mechanistic model handles the force-displacement relationship during interactions between the haptic device and 3D virtual objects. For the sake of simplicity, the mechanical impedance of the interactions between the simulated rigid objects and the generic stylus of haptic device is modeled by means of a simple spring law ($F = kx$). Following the detection of collision, we compute the distance between the collision point (i.e. ideal haptic interface point) and the tip of the simulated stylus (i.e. haptic interface point) and multiply this distance with the gross impedance of the object to compute the total reaction force. Then, for frictionless objects, the component of this force along the direction of surface normal at the collision point is calculated and sent as a force command to the force feedback device to provide the user with the tactual sensation of object shapes. This concept is depicted in Figure 2b. In this figure, A represents the haptic interface point, and P is the collision point which is on the surface of the polygon. The component of the vector (**AP**) along the surface normal ($\vec{N}_S$) is multiplied by stiffness coefficient (k) to compute the reaction force in the normal direction. Since the virtual objects can have only a finite stiffness, the haptic interface point will penetrate into the object until the reaction force prevents the user to move further.

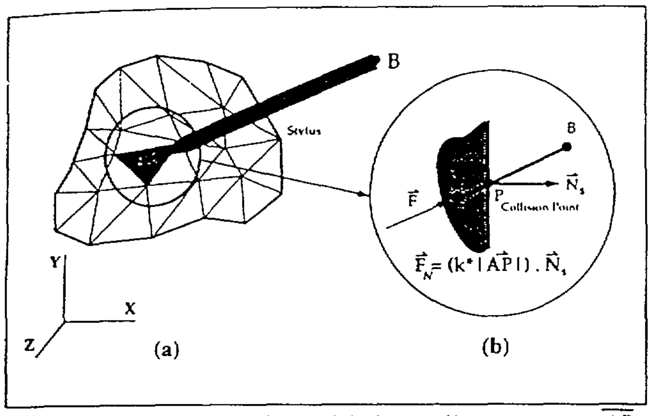


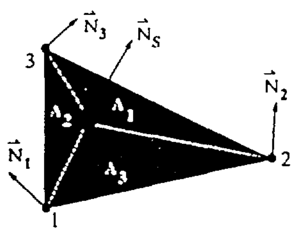
Figure 2. (a) The stylus is modeled as a line segment ($\overline{AB}$) for the purposes of fast collision computations. (b) Following the detection of collision, the reaction (contact) force is computed using the mechanistic model ($\vec{F} = k$ **AP**).

We have also successfully added static and dynamic frictional forces to the normal component of the reaction force in the direction tangential to the motion, by modifying the technique described in Salisbury et al. (1995). In the future, we plan to develop more sophisticated mechanistic models to handle various types of "instrument-object" interactions.

2.3 Haptic Shading and Filtering

Surface Smoothness: In computer graphics, illumination models are used to compute the surface color at a given point of a 3D object (Foley et al., 1995). Then, shading algorithms are applied to shade each polygon by interpolating the light intensities (Gourand shading) or surface normals (Phong shading). In addition to giving an impression of the three dimensionality of the object, shading algorithms make the shared edges of the adjacent polygons invisible and provide the user with the display of visually smooth surfaces (Watt and Watt, 1992). One can integrate a similar approach into the study of haptics to convey the feel of haptically smooth object surfaces. It is well known that our tactile sensory system is sensitive to edges and curvatures (Srinivasan and LaMotte, 1991) of objects. Hence, 3D geometrical objects which are constructed from polygons and look smooth to our eyes on the computer screen do not feel smooth haptically due to discontinuities in force and geometry. To overcome this problem, Morgenbesser and Srinivasan (1996) suggested the idea of *force shading*, analogous to Phong shading in visual displays. Force Shading refers to a controlled variation in the direction of the force vector to minimize or eliminate the edge effects during haptic rendering of polygonal surfaces. These modified forces are then reflected to the user through a haptic interface device. This method with different force interpolation techniques has been used by Ruspini et al. (1996) and Fukui (1996) to display object shape.

We improved the smoothing technique originally suggested by Morgenbesser and Srinivasan, to render 3D objects. The proposed technique satisfies the boundary conditions and provides a more uniform haptic smoothing. During the pre-computation phase of our simulations, we compute the surface normal at each vertex by averaging the surface normals of neighboring polygons, weighted by their neighboring angles. During the real-time computations, we first detect the collision point which divides the collided triangle into three sub-triangles. Then, the normal vector at the point of contact ($\vec{N}_S$) can be calculated by using an area based interpolation as follows;



$$\vec{N}_S = \frac{\sum_{i=1}^{3} A_i \cdot \vec{N}_i}{\sum_{i=1}^{3} A_i} \qquad (1)$$

where, $\vec{N}_i$'s are the pre-computed vertex normals of the collided triangle, and A_i's are the areas of the sub-triangles respectively.

Haptic texturing:
Haptic texturing is a method of simulating surface properties of objects in virtual environments in order to provide the user with the feel of macro and micro surface details. Several texturing techniques have been described by other authors (Siira and Pai, 1996; Plesniak and Underkoffler, 1996; Fritz and Barner, 1996).

<u>Approach</u>: Our approach is to haptically render the virtual objects using the "bump mapping" technique of computer graphics to provide the user with the sense of surface details such as bumps and textures. Bump mapping is a well known graphical technique for generating the appearance of a non-smooth surface by perturbing the surface normals. This

technique was initially proposed by Blinn (1978) as a method of graphically rendering 3D bumps for modeling clouds. His formulation depends on parameterization of the surface. Although the surface parameterization techniques have some advantages, they may generate uneven bump textures on the surface of complex shapes. Max and Becker (1994) suggested a direct method of mapping that does not require a transformation from global to parametric space. They developed a formulation which is purely based on the original surface normal and the local gradient of the height field that generates bumps on the surface of the object. Max and Becker utilized this technique to generate bumps on cloud contour surfaces. We used the same approach and perturbed the direction and magnitude of the surface normal ($\bar{N}_s$) to generate bumpy or textured surfaces that can be sensed tactually by the user in virtual environments. The perturbed surface normals ($\bar{M}$) can be computed using the following formulation;

$$\bar{M} = \bar{N}_s - \nabla h + (\nabla h \cdot \bar{N}_s)\bar{N}_s \qquad (2)$$

$$\nabla h = \frac{\partial h}{\partial x}\hat{i} + \frac{\partial h}{\partial y}\hat{j} + \frac{\partial h}{\partial z}\hat{k} \qquad (3)$$

where, h(x,y,z) represents the height (texture) field function and ∇h is the local gradient vector

__Implementation:__ In order to simulate haptic textures in virtual environments, a minimum of two parameters are needed;

- a height descriptor
- a frequency descriptor.

We also need to know the spatial and temporal characteristics (e.g haptic resolution) of the force feedback device, as well as the human finger, to properly set these descriptors in our simulations. Keeping these in mind, we have ported the texturing techniques of computer graphics to simulate haptic textures. The developed haptic texturing techniques can be classified into two parts: (a) image-based and (b) procedural.

__(a) *Image-based haptic texturing*__: deals with constructing a texture field from a 2D image data. In computer graphics, the digital images are wrapped around 3D objects to make them look more realistic. However, the graphical texture map contains only 2D color or gray scale intensities which do not necessarily correspond to the height intensities of the object surface. In haptic texturing, the goal is to make the user feel the height variations of the texture. For this reason, we propose *haptel* as a new element for simulating image-based haptic textures. Haptel is the smallest element of the haptic texture and is simply a texel (the smallest element of the graphical texture) with a height value. Haptels, when assigned to each texel, will automatically generate a discrete height field in t texture coordinate system. For example, the gray sc intensity of each texel can be used directly as a heig indicator to generate a haptic texture field.

Now, the goal is to map the height field of the digital ima onto a 3D polygonal object so that we can associate a textu coordinate (u,v) with the coordinate of each vertex (x,y,z). V map the height intensities of the digital image onto the obje surface using the two-stage texture mapping techniques computer graphics (Bier and Sloan, 1986; Watt and Wa 1992). The first stage is mapping from 2D texture space to simple intermediate surface such as a plane, cube, cylinder, sphere. The second stage provides the mapping from tl intermediate surface to the object surface which can l achieved in various ways (Watt and Watt, 1992). W experimented with a plane and a sphere as our intermedia surfaces and were able to map a digital image of a "brick surface onto a 3D object surface using the two-stage mappir technique (see Figure 3a and refer to Appendix A fo implementation details).

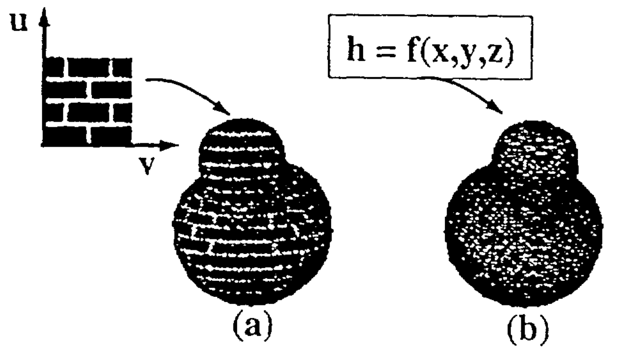


Figure 3. Visual display of textured surfaces that can b sensed via a haptic device. (a) A "brick" texture is mappe onto a 3D object and its gray scale values are used to generat a height map that can be sensed tactually. (b) Procedura haptic texturing techniques are used to map the filtered whit noise function onto a 3D object.

Once the mapping is achieved, then the aim is to compute the local gradient of the discrete height field at a given objec coordinate. We first determine the collision point and its neighboring points which are on the object surface and within a small distance of epsilon (ε). Then, we compute the components of the local gradient vector using the central difference approximation for partial derivatives:

$$\frac{\partial h}{\partial x} = \frac{(h_{x+\varepsilon} - h_{x-\varepsilon})}{2\varepsilon} \qquad (4)$$

$$\frac{\partial h}{\partial y} = \frac{(h_{y+\varepsilon} - h_{y-\varepsilon})}{2\varepsilon} \qquad (5)$$

$$\nabla h = \frac{\partial h}{\partial x}\hat{i} + \frac{\partial h}{\partial y}\hat{j} \qquad (6)$$

nce the local gradient is known, equation (2) can be used to erturb the surface normal at the collision point for simulating mage-based textures.

(b) *Procedural haptic texturing*: The goal of the procedural haptic texturing is to generate synthetic textures using mathematical functions. We used Fourier series, stochastic functions (e.g. filtered white noise), and a combination of these mathematical functions with fractals to generate synthetic texture fields.

To model clouds graphically, Gardner (1985) used Fourier series with four to seven sine terms according to the following formulation:

$$h(x,y) = k\sum_{i=1}^{n}[C_i \sin(f_i x + p_i) + T_o]\sum_{i=1}^{n}[C_i \sin(g_i y + q_i) + T_o] \qquad (7)$$

where, f_i, g_i are the fundamental frequencies and p_i, q_i are the phase frequencies in the x and y directions, C_i are the amplitudes of the texture at various frequencies, k and T_o are scalar constants. We applied the above method to haptically display periodic textured surfaces. (refer to Appendix B for more details). Once the mathematical function $h(x,y)$ is defined, such as in Eq.(7), the gradient vector (∇h) at the point of contact can be computed easily by differentiating the terms of the function with respect to x and y and inserting the coordinates of the collision point. Then, this vector can be utilized to perturb the surface normal based on the formulation given by Max and Barker (1994). We have applied Fourier series approach to haptic texturing and observed that several different textures could be generated by modifying the frequencies and coefficients of the Eq. (7). However, the variations decrease as the number of terms increase. Moreover, the haptic resolution (temporal and spatial characteristics) of the human finger does not precisely match that of the PHANToM, hence, the changes made on the direction and magnitude of the surface normal may need to be amplified to compensate this difference. If the generic probe of the haptic device is not capable of sensing at the level of human resolution, then the actual amplitude of the bumps used in simulations must be a number of times higher than the one computed from an analytical function.

Many of the textures in nature do not have regular (periodic) patterns. Textures such as sand, grass, fur, rocks, water surface cannot be modeled easily using the Fourier series. However, it has been shown that the height of many irregular surfaces follow Gaussian distribution and can be modeled using stochastic functions. One of the well known stochastic functions is white noise. A pseudo-random generator is a good source for generating white noise. Perlin (1985) defined a special kind of stochastic function known as "Noise(x,y,z)" to generate 3D synthetic graphical textures (also known as solid textures). In order to create a solid texture, a 3D cubical lattice is generated in texture space and pseudo-random numbers are assigned to each point of the lattice. This can be achieved by using a simple lookup table or a hashing function (Ebert et al., 1994). Then, the coordinates of the object are mapped to these lattice points in order to identify which lattice cell we are in for a given coordinate. Finally, the noise value at this coordinate is calculated by interpolating the pseudo-random values at the 8 nearby cell points. Thus, Noise(x,y,z) function takes a three dimensional vector and returns a scalar interpolated value. However, we need to know the gradient of the noise function at the point of contact for the proper simulation of haptic textures. Perlin (1985) defines a differential, DNoise(x,y,z), that is the gradient of Noise(x,y,z). Hence, DNoise(x,y,z) takes the coordinates of the collision point and returns a pseudo-random gradient vector (see Figure 3b and refer to Appendix C for implementation details). Once the gradient vector is known, the surface normal can be perturbed to generate haptic textures.

Another approach for constructing haptic textures is to use fractals along with mathematical functions such as Fourier series or noise. Fractals are also suggested for modeling natural forms since many natural objects seem to exhibit self-similarity (Mandelbrot, 1982; Ebert et al., 1994). For example, mountains have peaks and each peak contains smaller peaks in it. If this branching process repeats itself many times and the entire object becomes similar to its subportions, then the object is said to be self-similar (Foley, 1995). In order to generate haptic textures using fractals, we need a coarsely approximated model of the texture surface, then we can recursively subdivide height and frequency descriptors in the original model until the desired level of detail is obtained. In Ebert et al. (1994), Musgrave refers to this underlying model as the basis function (e.g. Fourier series, noise). For example, Musgrave uses Perlin's Noise(x,y,z) function to generate various textures graphically. He consecutively scales down the amplitude and spatial frequency of the bumps, generated by Noise(x,y,z) function, and adds them to the original shape. If this process is repeated several times, we end up with fractal surfaces. In haptic texturing, optimal value of this number depends on the spatial and temporal resolution of the human and the haptic device.

3. Discussion

We have developed generic algorithms which work with force reflecting haptic interfaces and enable the user to feel the forces that arise from interactions between simulated instruments and virtual objects. Our algorithms provide the user with computer controlled forces for tactual sensing of shape and surface details while they interact with rigid objects in virtual environments. Although these algorithms were implemented on the PHANToM, they are general enough to work with other types of force feedback devices that enable the user to feel the forces that arise from interactions between a variety of simulated tools and objects. Currently, many of the haptic devices cannot handle interaction torques. However, more sophisticated ones will be available in the future and the haptic rendering algorithms proposed here can be generalized to enable the user to feel interaction torques and object compliances. Many of the proposed techniques in this paper require more testing to validate their utility with various shapes and textures. Interested readers can refer to the Proceedings of the First PHANToM Users Group Workshop (Salisbury and Srinivasan, 1996) for more information on PHANToM-based haptics. We believe that more comprehensive studies which explore the haptic characteristics of humans and machines are required to characterize and render 3D objects in multimodal virtual environments.

Acknowledgments

Part of this work was carried out under the contract N61339-96-K-0002 funded by ONR and NAWC/TSD.

References

Bier, E.A., Sloan K.R., 1986, "Two-Part Texture Mapping", *IEEE Computer Graphics and Applications*, September, pp. 40-53.

Blinn, J.F., 1978, "Simulation of Wrinkled Surfaces", *ACM (Proceedings of SIGGRAPH)*, Vol. 12, No.3, pp. 286-292.

Ebert, D.S., Musgrave F.K., Peachey D., Perlin K., Worley S., 1994, "*Texturing and Modeling*", AP Professional, Cambridge, MA.

Foley, J.D., van Dam, A., Feiner, S. K., Hughes, J.F., 1995, "*Computer Graphics: Principles and Practice*", Addison-Wesley.

Fritz and Barner, 1996, "Haptic Scientific Visualization", in Proceedings of the First PHANToM Users Group Workshop, Eds: Salisbury J.K. and Srinivasan M.A. MIT-AI TR-1596 and RLE TR-612.

Fukui Y., 1996, "Bump mapping for a force display", in Proceedings of the First PHANToM Users Group Workshop, Eds: Salisbury J.K. and Srinivasan M.A. MIT-AI TR-1596 and RLE TR-612.

Gardner, G.Y., 1985, "Visual Simulation of Clouds", *ACM (Proceedings of SIGGRAPH)*, July, pp. 297-303.

Mandelbrot, B., 1982, "*The Fractal Geometry of Nature*", W.H. Freeman.

Massie T.H., Salisbury J.K., 1994, "The PHANToM H Interface: A Device for Probing Virtual Obj *Proceedings of the ASME Dynamic Systems and C Division*, DSC-Vol. 55-1, pp. 295-301.

Max, N.L., Becker, B.G., 1994, "Bump Shading for V Textures", *IEEE Computer Graphics and App.*, July, 14, pp. 18-20

Morgenbesser, H.B., Srinivasan, M.A., 1996, "Force Sh for Haptic Shape Perception", *Proceedings of the A Dynamic Systems and Control Division*, DSC-Vol. 5 407-412.

Perlin, K., 1985, "An Image Synthesizer", *ACM SIGGR.* Vol. 19, No. 3, July, pp. 287-296.

Plesniak and Underkoffler, 1996, "SPI Haptics Libr Proceedings of the First PHANToM Users C Workshop, Eds: Salisbury J.K. and Srinivasan M.A. AI TR-1596 and RLE TR-612

Ruspini, D.C., Kolarov, K., Khatib O., 1996, "Robust H Display of Graphical Environments", in Proceedin the First PHANToM Users Group Workshop, Salisbury J.K. and Srinivasan M.A. MIT-AI TR-1596 RLE TR-612.

Salisbury, J.K., Srinivasan, M.A., 1996, Proceedings o First PHANToM Users Group Workshop, Sept. 2 MIT Endicott House, Dedham, MA. (RLE TR-612, also available from http://www.ai.mit.edu/publications

Salisbury, J.K., Brock, D., Massie, T., Swarup, N., Zille 1995, "Haptic Rendering: Programming touch interac with virtual objects", *Proceedings of the ACM Sympo on Interactive 3D Graphics*, Monterey, California.

Siira, J., Pai D. K., 1996, "Haptic Texturing - A Stoch Approach", *Proceedings of the IEEE Internati Conference on Robotics and Automation*, Minneap Minnesota, pp. 557-562.

Srinivasan, M.A., LaMotte R.H., 1991, "Encoding of shap the responses of cutaneous mechanoreceptors",Informa *Processing in the Somatosensory System*, Ed.; O. Fran and J. Westman, pp. 56-69, Macmillan Press.

Srinivasan, M.A., 1995, "Haptic Interfaces", In *Vir Reality: Scientific and Technical Challenges*, Eds: Durlach and A. S. Mavor, pp. 161-187, National Acad Press.

Srinivasan, M.A., Basdogan, C., 1997, "Haptics in Vir Environments: Taxonomy, Research Status, Challenges", Computers and Graphics (in press).

Watt, A., Watt, M., 1992, "*Advanced Animation Rendering Techniques*", Addison-Wesley, NY.

Zilles, C.B., Salisbury, J.K., 1995, "A Constraint-based G object method for haptic display", *IEEE Internatio Conference on Intelligent Robots and System.*

Appendix A

Image-based haptic texturing that does not depend on shape of the object, known as two-stage mapping, is descri in detail in Bier and Sloan (1986) and Watt and Watt (199

Here, we briefly describe the steps of mapping a 2D texture image onto a 3D intermediate surface such as sphere, and then onto the object surface (see Figure 4).

- Use the "bounding box" coordinates of the object to calculate the center point, C(x,y,z).
- Calculate the collision point, P(x,y,z) using the ray-based collision detection algorithm.
- Map the object coordinates to spherical coordinates such that we can associate the point P(x,y,z) with a point on a unit sphere, $S(r = 1.0, \theta, \phi)$.

where,

$0 \leq \theta \leq 2\pi$ and $0 \leq \phi \leq \pi$

- Map the spherical coordinates to image coordinates (u,v) such that we can associate the point S with a height value (i.e. haptel).

$$(u,v) = \left[\frac{\theta}{2\pi}, \frac{\phi}{\pi}\right] \qquad u, v \in [0,1] \qquad \text{(A.1)}$$

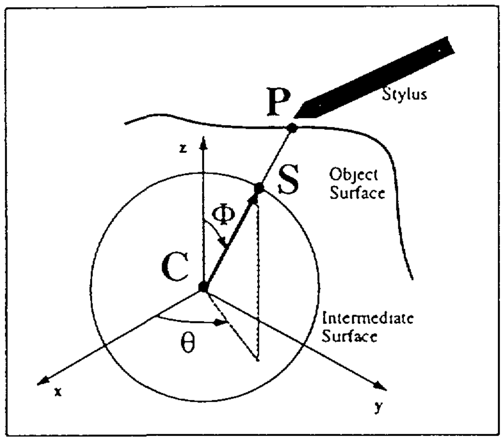


Figure 4. Schematic of mapping 2D images onto 3D surfaces for generating image-based haptic textures: Following the computation of the collision point (point P), we compute its equivalence in spherical coordinates (point S) using the center coordinates of the bounding box that encloses the object (point C). We finally define a mapping function between the image coordinates (u,v) and the spherical coordinates to associate the point S with a height value.

Appendix B

Fourier series: Gardner(1985) suggests that natural looking texture patterns can be produced if the frequencies and coefficients of Eq. (7) are chosen by relationships:

$$f_{i+1} = 2f_i$$
$$g_{i+1} = 2g_i$$
$$C_{i+1} = \frac{\sqrt{2}}{2}C_i \qquad \text{(B.1)}$$

Moreover, Gardner increases the texture variations by shifting the sine component in the x direction as a function of y and the y component as a function of x using the following relations

$$p_i = (\frac{\pi}{2})\sin(\frac{g_i y}{2}) \qquad i > 1$$
$$q_i = (\frac{\pi}{2})\sin(\frac{f_i x}{2}) \qquad i > 1 \qquad \text{(B.2)}$$

Appendix C

DNoise function takes the coordinates of the collision point and returns a pseudo-random vector In Ebert et al. (1994), Perlin describes the steps of generating graphical textures using the *noise* function. Here, we briefly describe the algorithm for the convenience of the reader. More details can be found in Ebert et al. (1992) and Perlin (1985).

- Generate a table of "256" pseudo-random unit vectors, G[256][3].
- Generate a column of "256" pseudo-random integers, M[256], varying from 0 to 255 such that each number occurs only once.
- Calculate the collision point, P(x,y,z) using the ray-based collision detection algorithm.
- Adjust the spatial frequency of the noise signal (x = x/wavelength; y = y/wavelength; z = z/wavelength)
- Map the coordinates of the collision point P(x,y,z) to the lattice cells and compute the collision coordinates P(u,v,w) relative to the cubic cell that we are in (see Figure 5).

$$i = floor(x); j = floor(y); k = floor(z);$$
$$P(u,v,w) \equiv (x-i, y-j, z-k) \qquad \text{(C.1)}$$

where,

i, j, k represent the indices of the *lowest* corner of the cell in which we are in.

- Use the column vector M to index into the table G for the each corner of the cell.

$$G_{i,j,k} = G[\text{mod}(\text{mod}(\text{mod}(M[i],256) + M[j],256) + M[k],256)][3] \qquad \text{(C.2)}$$

- Assume a weight function, $\Omega(u,v,w)$, where the variation of weight for each component is given by a cubic approximation. Smoothly interpolate the noise gradient assigned to each corner of the cubic cell with the weight function such as

$$\Omega(u,v,w) = (1-3u^2+2u^3)*(1-3v^2+2v^3)*(1-3w^2+2w^3) \tag{C.3}$$

- Calculate the derivative of noise as the summation of eight corners of the cell by taking into account the pseudo-random vector at each corner, weight function, and the distance from each corner of the cell to P(u,v,w).

$$\begin{aligned} Noise = \{ & \Omega(u,v,w)(G_{i,j,k}) + \Omega(1-u,v,w)(G_{i+1,j,k}) + \\ & \Omega(1-u,1-v,w)(G_{i+1,j+1,k}) + \Omega(1-u,1-v,1-w)(G_{i+1,j+1,k+1}) + \\ & \Omega(u,1-v,1-w)(G_{i,j+1,k+1}) + \Omega(u,v,1-w)(G_{i,j,k+1}) + \\ & \Omega(1-u,v,1-w)(G_{i+1,j,k+1}) + \Omega(u,1-v,w)(G_{i,j+1,k}) \} \end{aligned} \tag{C.4}$$

$$DNoise = (\frac{\partial}{\partial u} Noise)\hat{i} + (\frac{\partial}{\partial v} Noise)\hat{j} + (\frac{\partial}{\partial w} Noise)\hat{k} \tag{C.5}$$

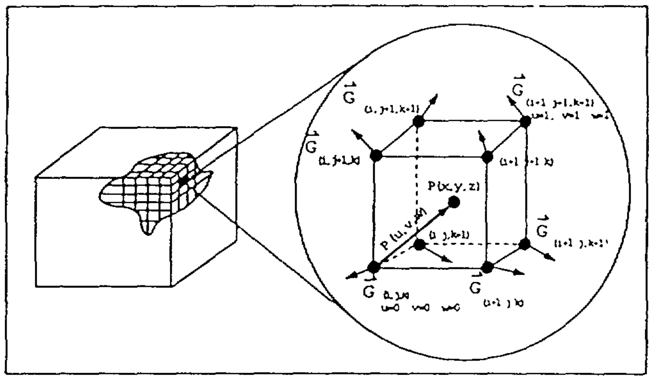


Figure 5. Procedural haptic texturing (implementation of noise textures): The space that surrounds the 3D object is divided into a lattice of cubical cells and a unit pseudo-random vector is assigned to each corner of the cell. Following the detection of the collisions, we first compute which cubical cell we are in. Then, the derivative of the noise at the collision point is calculated as the derivative of the weighted sum of the distances of each corner from the collision point times the pseudo-random vectors assigned to the corners of the cell.