Haptic Rendering of Complex Objects via Directional Sampling

Huagen Wan[1*] Xiaoxia Han[2**] Zhihua Zhou[1†]
[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, China
[2]College of Information Sciences and Electric Engineering, Zhejiang University, Hangzhou, 310027, China

Abstract

Haptic rendering is a growing concern of research community worldwide. However, it is a great challenge to haptically render complex objects. In this paper, a novel haptic rendering algorithm via directional sampling is proposed. We propose haptic box, an intermediate representation of target objects for haptic rendering, which encodes the intersection data of each sample directional ray with the target object. With haptic box, the contact information between a line segment and the target object can be derived by several line-box intersection tests and bilinear interpolations, and can be directly used to evaluate the feedback force. Experimental results show that our algorithm is insensitive to model complexity at the run-time stage and works both effectively and efficiently.

CR Categories: H.5.2 [User Interfaces]: Haptic I/O, I.3.5 [Computational Geometry and Object Modeling]: object representations

Keywords: haptic rendering, collision detection, force feedback, directional sampling

1 Introduction

Haptic rendering is a process by which users feel the virtual objects. Many haptic rendering algorithms have been proposed in past decades, yet it is challenging to haptically render complex geometry objects [Duriez et al. 2006; Laycock et al. 2007; Srinivasan et al. 1997]. A complex geometric model usually contains tens of thousands, or even, millions of polygons. It is very difficult to achieve update frame rate as high as 1000 HZ for most haptic rendering algorithms which are sensitive to model complexity. The bottleneck lies in collision detection and proximity query since it is almost the most expensive process in haptic rendering. For over half a century, the problem of collision detection has been intensively investigated and a lot of mile-stone algorithms have been proposed [Jimenez et al. 2001; Lin et al. 1998; Pascale et al. 2006]. However, there are still big gaps between the performance of collision detection algorithms and the requirements of haptic rendering.

[*]e-mail: hgwan@cad.zju.edu.cn
[**]e-mail: hanxx@ zju.edu.cn
[†]e-mail: zhouzhihua@cad.zju.edu.cn

VRCAI 2008, Singapore, December 8 –9, 2008.
© 2008 ACM 978-1-60558-335-8/08/0012 $5.00

In this paper, we present a novel haptic rendering algorithm for complex objects for applications such as computer games where accuracy is not of great concern. We propose the concept of haptic box, which is an intermediate representation of target objects. It encodes the intersection data of each sample directional ray with the target object. With haptic box, contact information between a line segment and the target object can be derived by several line-box intersection tests and bilinear interpolations, and can be directly used to evaluate the feedback force.

2 Related Work

During the past decade, much effort has been made to develop haptic rendering algorithms for various haptic devices. These methods can be classified into categories according to the avatars used: point-based methods, ray-based methods and object-based methods. A typical point-based method simulates the generic probe of a haptic device as a single point. Therefore, the problem is simplified to render the interaction between a single point and target models [Ruspini et al. 1997]. In ray-based methods, a directional line segment is used as the avatar of the haptic input and the interferences are queried between the line segment and the target objects [Basdogan et al. 1997; Ho et al. 2000]. Object-based methods directly render the interaction between two 3D objects and particular attentions are paid on how to reduce the computational complexity through convex decomposition [Ehmann et al. 2001; Gregory et al. 2000] or spatial coherence [Johnson et al. 2004; Kim et al. 2003; Park et al. 2004].

Though many algorithms have been proposed, it is still challenging to haptically render complex geometries since most algorithms are sensitive to model complexity. Nowadays, it is well established that the problem of proximity query has become a bottleneck in haptic rendering. To solve this problem, different representational schemes of 3D objects have been investigated [McNeely et al. 1999; Otaduy et al. 2003; Pascale et al. 2006; Shahram et al. 2005].

To meet the requirement of high update rate of PHANTOM, McNeely et al. propose a complexity-controllable method named Voxmap-PointShell. All geometric objects are discretized into voxels or points according to whether they were static or dynamic [McNeely et al. 1999]. In [Otaduy et al. 2003] and [Shahram et al. 2005], the concept of Level-of-Detail (LOD) was introduced for haptic rendering. Pascale and Prattichizzo propose a framework for bounded time collision detection for point-like haptic interactions. The so-called V-GRAPH framework employs strategies similar to those used by the Lin-Canny and Dobkin-Kirkpatrick algorithms [Lin et al. 1991; Dobkin et al. 1993] but, differently from these ones, it uses a partition of the space focused on vertices only, which allows both for an easier implementation

and for usage with non-convex objects without the need for splitting the original mesh [Pascale et al. 2006].

Our haptic rendering algorithm is based on directional sampling. We propose haptic box, a new intermediate representation of target object to facilitate haptic rendering of complex geometries. The haptic box stores the intersection data of each sample directional ray with the target object. In fact, it is quite common to use simple shapes to store information in computer graphics. For instances, spheres are used in environment mapping [Blinn et al. 1976], hemi-cubes are used in radiosity method [Cohen et al. 1985], and light field mapping is used for rendering surface light fields [Chen et al. 2002]. These facinating works inspired us to develop our haptic box representation.

3 Algorithm Overview

The haptic rendering algorithm we propose works in two stages: preprocessing stage and run-time stage (Figure 1). It belongs to the category of ray-based methods where directional line segments are used as avatars of haptic input interfaces.

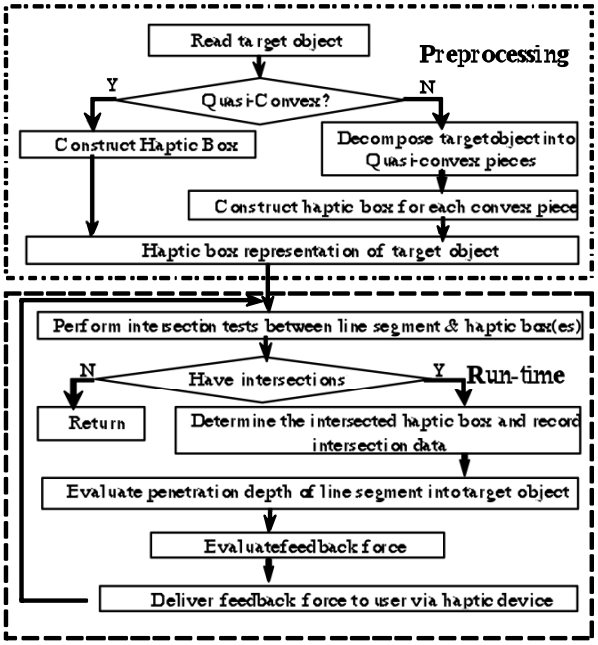


Figure 1. Framework of our algorithm.

In the preprocessing stage, a haptic box is or haptic boxes are first constructed for a target object, depending upon the structure of the target object. Then the haptic box(es) is/are used as a substitute of the original target model. In the run-time stage, intersection test is first performed between a line segment and the haptic box(es), and the penetration depth of the line segment into the target object can be directly derived from the intersection test results, then the feedback force is evaluated, and finally the force is delivered to the user through a haptic feedback device.

4 Construction of Haptic Box via Directional Sampling

In the preprocessing stage, we first read the target object, then decompose it into quasi-convex pieces according to its structure, and finally construct a haptic box for each quasi-convex piece.

The construction of a haptic box for a quasi-convex piece consists of two steps. First, a positional bounding box (PBB) is constructed and each of the six faces of the PBB is uniformly partitioned into a grid to generate position points (e.g. $\boldsymbol{P_0}$ in Figure 2). The PBB is used for positional interpolation. Second, a directional hemi-cube (DHC) is constructed at each position point. DHCs are used to record information about intersections between line segments and target model. As illustrated in Figure 2, a haptic box consists of a PBB and DHCs at each position point. The large box drawn with black lines is the PBB, while the small hemi-cube drawn with red lines is a DHC at the position point $\boldsymbol{P_0}$. Before discussing details about construction of PBB and DHC, we first introduce the decomposition of the target model.

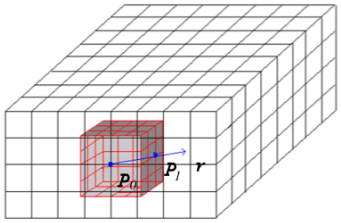
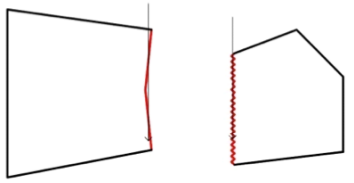


Figure 2 Haptic Box

4.1 Quasi-convex Decomposition

Mathematically, a function $f: S \to \Re$ defined on a convex subset S of a real vector space is quasi-convex if whenever $x, y \in S$ and $\lambda \in [0,1]$ then $f(\lambda x + (1-\lambda)y) \le \max(f(x), f(y))$ [Wiki 2008]. In this paper, we don't follow the above strict definition. Instead, we describe a quasi-convex object as a natural extension of a convex object. It shares similar shape as the convex object and can be rendered in a similar way that the convex one is rendered. In other words, the haptic rendering of a quasi-convex object is not affected by its concave features. Figure 3 shows two kinds of such concave features: flat feature and compact feature. In the former case, the rendering of one face of the flat feature will probably not be affected by another face. In the later case, the gaps would be imperceptible due to limited resolution of human touch organs if they are small enough [Wheat et al. 2000].



(a) Flat concave feature (b) Compact concave feature

Figure 3. Examples of quasiconvex object.

For a general concave object, we need first decompose it into quasi-convex pieces to facilitate the construction of haptic boxes. Convex or approximate convex decomposition methods have been extensively studied in computer graphics [Bajaj et al. 1992; Chazelle 1990; Chazelle 1997; Ehmann et al. 2001; Lien 2006], but they can not be directly used in our algorithm. Considering the difficulty of quasi-convex decomposition and eager to test our idea, we use manual decomposition at present time.

4.2 Construction of PBB

The PBB of a convex piece has the same size as the bounding box of quasi-convex piece. Its construction consists of three steps.

First, the bounding box of the quasi-convex piece is constructed. PBB is critical to our haptic rendering process. We think that a good PBB should not only be independent of the coordinate system but also be as tightly as possible. Following these two constraints, the oriented bounding box (OBB) is chosen as the bounding box of the quasi-convex piece [Gottschalk et al. 1996]. Second, each face of the bounding box is partitioned into a grid to get position points. The uniform grid is chosen due to its simplicity as well as for the purpose of fast positional interpolation. Finally, a DHC is constructed at each position point.

4.3 Construction of DHC

To construct a DHC at a position point of the PBB, a virtual hemi-cube is centered at the position point with each of its five faces being uniformly partitioned into a grid to generate directional points. A DHC serves for two purposes. First, it records the intersection information along sample directions. This is achieved by recording at directional points the intersections of line segments and the target model. Second, it is used for fast directional interpolation. Figure 4 illustrates a DHC centered at the position point $\boldsymbol{P_0}$. Each directional point of the DHC corresponds to a ray originating from $\boldsymbol{P_0}$ (For instance, $\boldsymbol{P_1}$ is a directional point of the DHC, and the ray $\overrightarrow{P_0P_1}$ can be represented by equation $L(t) = P_0 + t \times \overrightarrow{P_0P_1} / \left\| \overrightarrow{P_0P_1} \right\|, t > 0$).

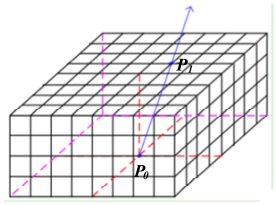


Figure4 Directional Hemi-Cube

The intersection information of the ray and the model is recorded at the directional point, including the parameter value t_0 at the intersection point, and the normal of the intersected triangle, etc.

4.4 Data Compression

The precision of our algorithm is affected by the sampling densities of PBB and DHCs. The more densely split the PBB and DHCs are, the more accurate the penetration depths are. However, the data that need be stored increase with the densities of the PBB and DHCs. The great amount of data would dramatically decrease the performance of our algorithm if they were not effectively handled.

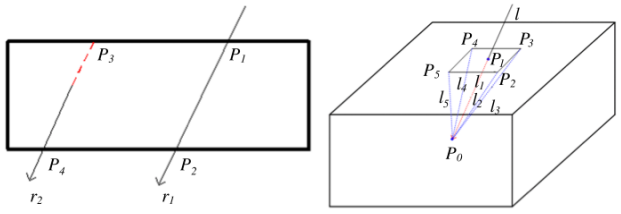
Data compression is the process of encoding information using fewer bits than an unencoded representation would use through use of specific encoding schemes. There are generally two categories of data compression: lossless and lossy compression. While typical lossless compression algorithms represent the data more concisely without error, typical lossy compression algorithms try to achieve larger compression rate if some loss of fidelity is acceptable. We choose Haar wavelet transform to compress DHC data due to its simplicity [Mulcahy 1996]. Other more advanced reasons for choosing it lie in following 3 aspects. First, though human haptic perception is sensitive to subtle variations in force, it allows force tolerances to some extent. Second, the memory cost of haptic box is generally huge even for moderate sampling densities of PBB and DHCs. While lossless compression algorithms can recover data without error, their compression rates are relatively too small to be usable in our algorithm. Third, haptic rendering requires a very high update frame rate. This demands that the uncompress process be performed on the fly. Moreover, haptic rendering is an interactive process, this requires that the uncompress algorithm should be able to randomly performed on any particular data.

5 Feedback Force Computation

We use directional line segments as avatars of haptic input interfaces. The haptic boxes are used as the substitute of the target object for fast computation. We first perform intersection tests between the line segment and all the haptic boxes. If there is no intersection between the line segment and any haptic box, the line segment does not intersect with the target object. Otherwise, the haptic box which intersects with the line segment is identified and the penetration depth of the line into the haptic box is recorded. Then the information stored in the haptic box is used to determine whether the depth value is large enough for the line segment to penetrate into the target object. If yes, the penetration depth of the line segment into the target object is evaluated by interpolations and directly used to calculate the feedback force. The details are introduced in the following.

5.1 Intersection Test between Line Segment and PBB

We are only interested in external intersection point. An external point is defined as the intersection point when a directional line segment enters the box from outside, while an internal intersection point is defined as the intersection point when the line segment leaves from inside the box. As indicated in Figure 5(a), the directional line segment $\boldsymbol{r_1}$ intersects the box at two points: $\boldsymbol{P_1}$ is an external intersection point, and $\boldsymbol{P_2}$ is an internal intersection point. The line segment $\boldsymbol{r_2}$ has only one internal intersection point $\boldsymbol{P_4}$, so we inversely extend the line segment to get the external intersection point $\boldsymbol{P_3}$. In Figure 5(b), the line segment $\boldsymbol{l}$ intersects with the PBB at external intersection point $\boldsymbol{P_1}$, and the penetration depth is $\|l_1\| = \left\| \overrightarrow{P_1P_0} \right\|$.



(a) External and internal intersection points (b) Bilinear positional interpolation

Figure 5 Intersection test between line segment and PBB and bilinear positional interpolation.

5.2 Bilinear Positional Interpolation

If the external intersection point $\boldsymbol{P_1}$ is at a position point of the PBB, we can easily retrieve the intersection information (e.g. t_0) of the ray originating from $\boldsymbol{P_1}$ along the direction $\boldsymbol{l_1}$ with the target object. If $t_0 < \| l_1 \|$, then the line segment intersects with the target object, and the penetration depth is $\| l_1 \| - t_0$. Otherwise, we can not directly find out t_0 (We can not even directly judge whether the line segment intersects with the target object). But fortunately, t_0 can be evaluated via bilinear positional interpolation.

As illustrated in Figure 5(b), we can easily get the grid $\boldsymbol{P_2P_3P_4P_5}$ where the external intersection point $\boldsymbol{P_1}$ locates. Connecting the position points $\boldsymbol{P_2}$, $\boldsymbol{P_3}$, $\boldsymbol{P_4}$, and $\boldsymbol{P_5}$ we get directional line segments

l_2, l_3, l_4, and l_5. Therefore, the intersection value t_0 of P_1 along the direction l_1 can be evaluated by bilinear positional interpolation:

$$t_0 = d_1 d_2 t_2 + (1-d_1) d_2 t_3 + d_1 (1-d_2) t_5 + (1-d_1)(1-d_2) t_4 \quad (1)$$

Where t_i are parameter values of the intersection points between l_i (i=2,3,4,5) and the target object which are pre-calculated and stored in the haptic box, d_1 is the ratio of the distance from P_1 to $\overrightarrow{P_2P_5}$ and $\overrightarrow{\|P_2P_3\|}$, and d_2 is the ratio of the distance from P_1 to $\overrightarrow{P_2P_3}$ and $\overrightarrow{\|P_2P_5\|}$.

5.3 Bilinear Directional Interpolation

Through bilinear positional interpolation, the problem of computing t_0 at any point P on the PBB along any direction is converted into the problem of finding t_0 at the four positional points of the grid where P locates. As DHCs only sample the direction at directional points, we use bilinear directional interpolation to compute t_0. To avoid time-consuming computation of anti-trigonometric functions, we convert the problem of directional interpolation into the problem of directional point interpolation.

Given a direction (x,y,z) and a DHC, the DHC is split into four sub-blocks by the plane xz and the plane yz (Figure 6). By limiting the intersection tests within a sub-block K (e.g. the orange one), the intersection point between r and K (e.g. P_1) is firstly found. Then the t_0 value along the direction (x,y,z) is evaluated by bilinearly interpolating the four t_0 values stored at the four directional points of the grid where the intersection point locates. As anti-trigonometric functions are avoided, the interpolation is very efficient.

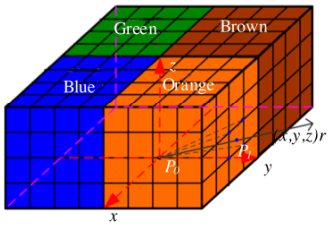


Figure 6 Directional interpolation.

5.4 Collision Query and Force Computation

If $t_0 > \| l_1 \|$, there is no intersections between the line segment and the target object, Otherwise, they may intersect, and if they intersect the penetration depth is $d = \| l_1 \| - t_0$. To determine where the line segment and the target object intersect, the originating point of the line segment should be considered. As shown in Figure 7, there are generally four possible configurations between the line segment and the target object. While the intersections in the configurations of r_3 and r_4 are valid, there is clearly no valid intersection between r_2 and the target object, and the case of r_1 does not exist in the real world, hence the intersection is also invalid.

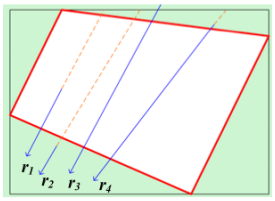


Figure 7. Intersection configuration

Assuming the length of the directional line segment is L, then the intersection between the line segment and target object is valid if and only if $\Delta = \| l_1 \| - t_0 - L < 0$.

To summarize, the collision query between a line segment and the target object can be fulfilled by several intersection tests between the line segment and the box, a bilinear positional interpolation and four bilinear directional interpolations. Therefore, our algorithm is very high efficient.

Once we query the penetration depth of the line segment into the target object and other information (e.g. polygon normal), the feedback force can be evaluated by the Hook's law.

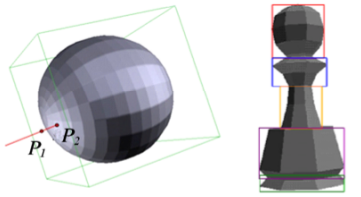
$$\vec{F} = k d_p \vec{N}_p \quad (2)$$

Where k is the object stiffness, $\vec{N}_p$ is the unit normal of the polygon at the intersection point, and

$$d_p = d \times (N_p, \frac{l_1}{\|l_1\|}) = (\| l_1 \| - t_0) \times (N_p, \frac{l_1}{\|l_1\|}) \quad (3)$$

6 Experiments and Discussions

The algorithm has been implemented using C++ and tested on a PC with Intel Pentium IV 2.8G CPU, 1G RAM. A complex quasi-convex sphere and a chess model are employed as the target objects for testing. The quasi-convex sphere has over 200,000 triangles (Figure 8(a)). The chess model has over 100,000 triangles, and is manually decomposed into 5 quasi-convex pieces with one haptic box built for each piece (Figure 8(b)).



(a) Quasi-convex sphere (b) Chess model

Figure8. Two testing models

6.1 Testing Results on Quasi-Convex Sphere

6.1.1 Efficiency Test

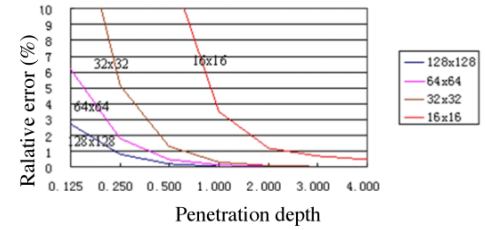
To perform efficiency test, we construct the haptic box of the quasi-convex sphere. There is a DHC at each position point of the PBB. The t_0 values stored at each directional point of a DHC are mapped into a 24-bit RGB image. The haptic rendering frame rate of our algorithm is about 41.5K frames/second, much higher than 1K frames/second, the frame rate required by typical haptic devices.

Analysis of our algorithm shows that the high efficiency is achieved by substituting the quasi-convex sphere with its haptic box, thus avoiding time-consuming collision queries. At run-time stage, our algorithm is independent on model complexity. In fact, collision queries between a line segment and the quasi-convex sphere involve only 1 intersection test between the line segment and the OBB of the quasi-convex sphere, 1 positional interpolation and 4 directional interpolations.

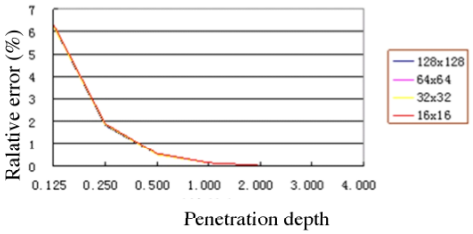
6.1.2 Precision Test

The algorithm precision affects its rendering effect. Our algorithm in its essence replaces the complex quasi-convex sphere with its haptic box, hence the penetration depth it evaluates is an approximate one to that of accurate algorithm. We perform precision tests on our algorithm as described below. The algorithm precision is measured by relative error

$\eta_0 = (|F_0 - F_1| / F_0) \times 100\%$, where F_1 is the feedback force evaluated by our algorithm, and F_0 is the force evaluated by accurate algorithm. We use different sampling densities and different penetration depths to test the precision of our algorithm. The test results are shown in Figure 9, in which the sampling density $m \times n$ means that the sampling density on each face of the PBB is $m \times m$ and the sampling density on each face of the four sub-blocks of each DHC is $n \times n$.



(a) Effect of PBB's sampling density on precision



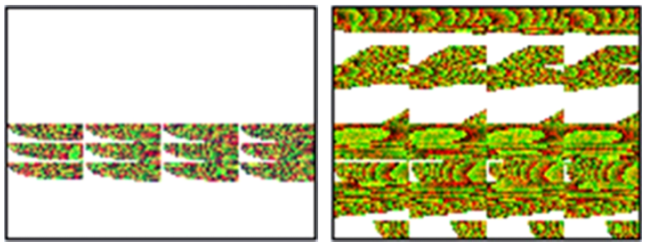
(b) Effect of DHCs' sampling density on precision

Figure 9. Precision test results.

6.1.3 Compression Test

We use Haar wavelet transform to compress DHC data. The aim of compression test is to test the compression rate *CR*, uncompress speed, and accuracy of the uncompressed data which are measured by relative error $\eta_1 = (|V_1 - V_0| / V_0) \times 100\%$, where V_0 and V_1 are original and uncompressed data value respectively.

Haar wavelet transform has different impacts on different DHC data. We take two typical DHCs as examples. In DHC A, there are no intersections at most directional points, while in DHC B, there are intersections at most directional points (Figure 10). Table 1 compares the compression results on DHC A and DHC B.



(a) DHC A (b) DHC B

Figure 10. Uncompressed DHC data stored as RGB images

Table 1. Compression results on DHC A and DHC B

	Case 1		Case 2		Case 3	
	CR	η_1	*CR*	η_1	*CR*	η_1
A	11.5:1	0.5%	15.5:1	1.5%	18.4	3%
B	4:1	0.5%	5.8:1	1.5%	7.3:1	3%

Figure 11 shows the relationship between compression rate and relative error, from which we can see that the Haar wavelet transform achieves good compression rate and uncompress effect on both DHC A and DHC B. Furthermore, the compression rate and uncompress effect on DHC A are better than those on DHC B. This is because the data of DHC A have better coherency than that of DHC B.

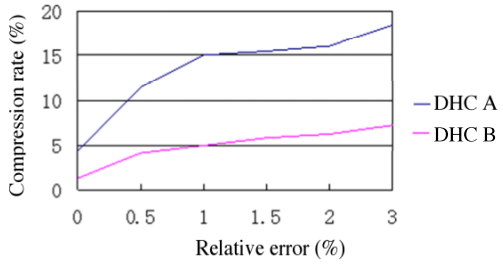


Figure 11. Relationship between compression rate and relative error.

The uncompress speed is relative to compression rate. For a compression rate of 8:1, it costs about 0.002ms to uncompress a value of t_0 in our tests.

6.2 Testing Results on Chess Model

We also performed tests on the chess model shown in Figure 8(b). For each piece, the testing results about precision and compression are the similar to those described in the section 6.1. We omit them due to limited space.

For efficiency tests, more than 1000 times tests were performed with the line segment being placed at different positions, different orientations and different penetration depths. Experimental results show that the haptic rendering frame rate is sometimes about 41.5K frames/second and sometimes about 20.7K frames/second, which are much higher than 1K frames/second, the frame rate required by typical haptic devices. Analysis shows that the variance in frame rate was caused by the number of haptic boxes the line segment intersects. In this testing case, the line segment intersects with at most two haptic boxes.

7 Conclusion

In this paper, a novel haptic rendering algorithm for complex objects is presented. Haptic box is proposed as a new representation of target objects for haptic rendering. As the contact information of sampled rays with target object is encoded into the haptic box, proximity query between a line segment and target object can be very efficiently achieved by several intersection tests between the line segment and the OBB(s) of the quasi-convex piece(s) of the target object, and bilinear interpolations. The query result can be directly used to evaluate feedback force.

Various tests, including efficiency test, precision test and compression test, are performed on a complex quasi-convex sphere and concave chess model with positive test results being achieved. At the run-time stage, our algorithm is insensitive to target model complexity. In practice, the algorithm runs with very high efficiency. Therefore, the proposed algorithm can be used in applications which need bounded time haptic interactions but not restrict on the precision of feedback force.

In the future, we will try to develop algorithms for automatically decompose the objects into quasi-convex pieces. Our basic idea is to build an OBB tree and merge from bottom up around the leaf nodes according to some quasi-convexity parameters like angle between two adjacent faces.

Acknowledgements

The authors would like to thank the 863 High Technology Plan of China (2006AA01Z130), and the National Natural Science Foundation of China (60673197) for financially supporting this research.

References

BAJAJ, C. AND DEY, T.K. 1992. Convex decomposition of polyhedra and robustness, *SIAM Journal on Computing, 21*, 339–364.

BASDOGAN, C., HO, C.H. AND SRINIVASAN, M.A. 1997. A ray-based haptic rendering technique for displaying shape and texture of 3d objects in virtual environments. In *Proceedings of ASME Dynamic Systems and Control Division 61*, 77-84.

BLINN, J. AND NEWELL, M. 1976. Texture and reflection in computer generated images. *Communications of the ACM*, 19, 10, 542-547.

CHAZELLE, B. AND PALIOS, L. 1990. Triangulating a non-convex polytope. *Discrete Computational Geometry 5*, 505–526.

CHAZELLE, B. AND PALIOS, L. 1997. Decomposing the boundary of a nonconvex polyhedron. *Algorithmica 17*, 245–265.

CHEN, W., BOUGUET, J., CHU, M., AND GRZESZCZUK, R. 2002. Light field mapping: efficient representation and hardware rendering of surface light fields. In *Proceedings of ACM Siggraph*, 447-456.

COHEN, M. AND GREENBERG, D. 1985. The hemi-cube: a radiosity solution for complex environments. In *Proceedings of ACM Siggraph,* 31-40.

DOBKIN, D., HERSHBERGER, J., KIRKPATRICK, D. AND SURI, S. 1993. Computing the intersection-depth of polyhedra. *Algorithmica 9*, 6, 518-533.

DURIEZ, C., DUBOIS, F., KHEDDAR, A. AND ANDRIOT, C. 2006. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics 12,1*, 36-47.

EHMANN, S. AND LIN, M. 2001. Accurate and fast proximity queries between polyhedra using convex surface decomposition. In *Proceedings of Eurographics*, 500-510.

GOTTSCHALK, S., LIN, M. AND MANOCHA, D. 1996. OBB-Tree: A hierarchical structure for rapid interference detection. In *Proceedings of the ACM Siggraph*, 171-180.

GREGORY, A., MASCARENHAS, A., EHMANN, S., LIN, M. AND MANOCHA, D. 2000. Six degree-of-freedom haptic display of polygonal models. In *Proceedings of .IEEE Visualization*, 139-146.

HO C., BASDOGAN C. AND SRINIVASAN M.A. 2000. Modeling of force and torque interactions between a line segment and triangular surfaces for haptic display of 3D convex objects in virtual and teleoperated environments. *International Journal of Robotics, 19, 7*, 668-684.

JIMENEZ, P., THOMAS, F. AND TORRAS, C. 2001. 3D collision detection: a survey. *Computers and Graphics 25, 2*, 269-285.

JOHNSON, D. E. AND WILLEMSON, P. 2004. Accelerated haptic rendering of polygonal models through local descent. In *Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 18-23.

KIM, Y., OTADUY, M., LIN, M. AND MANOCHA, D. 2003. Six degree-of-freedom haptic rendering using incremental and localized computations. *Presence: Teleoperators and Virtual Environments 12, 3*, 277-295.

LAYCOCK, S.D. AND DAY, A.M. 2007. A survey of haptic rendering techniques. *Computer Graphics Forum 26, 1*, 50-65.

LIN, M. AND CANNY, J. 1991. Efficient algorithms for incremental distance computation. In *Proceedings of IEEE Conference on Robotics and Automation*, 1008-1014.

LIEN J.M. 2006. Approximate convex decomposition and its applications. Ph.D. Thesis, Department of Computer Science, Texas A&M University. December 2006.

LIN, M. AND GOTTSCHALK, S. 1998. Collision detection between geometric models: A survey. In *Proceedings of IMA Conference on Mathematics of Surfaces*, 37-56.

MCNEELY, W.A., PUTERBAUGH, K.D. AND TROY, J.J. 1999. Six degree-of-freedom haptic rendering using voxel sampling. In *Proceedings of ACM Siggraph*, 401-408.

MULCAHY, C. 1996. Image compression using the Haar wavelet transform. *Spelman Science and Math Journal 1*. 22-31.

OTADUY, M.A. AND LIN, M. 2003. Sensation preserving simplification for haptic rendering. In *Proceedings of ACM Siggraph*, 543-553.

PARK, J. G. AND NIEMEYER, G. 2004. Haptic rendering with predictive representation of local geometry. In *Proceedings of the 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, 331-338.

PASCALE, M. DE AND PRATTICHIZZO, D. 2006. A framework for bounded-time collision detection in haptic interactions. In *Proceedings of the ACM symposium on Virtual reality software and technology*, 305-311.

RUSPINI, D.C., KOLAROV, K. AND KHATIB, O. 1997. The haptic display of complex graphical environments. In *Proceedings of ACM Siggraph,* 345-352.

SHAHRAM, P., JOHN, D. AND JIAN, Z. 2005. A study of level-of-detail in haptic rendering. *ACM Transactions on Applied Perceptions 2*, 15-34.

SRINIVASAN, M. AND BASDOGAN, C. 1997. Haptics in virtual environments: taxonomy, research status, and challenges. *Computers and Graphics 21, 4*, 393-404.

WHEAT, H.E. AND GOODWIN, A.W. 2000. Tactile discrimination of gaps by slowly adapting afferents: effects of population parameters and anisotropy in the fingerpad. *The Journal of Neurophysiology*, 84, 3, 1430-1444.

WIKI. 2008. http://en.wikipedia.org/wiki/Quasiconvex_function, last visit: August 25, 2008.