Multi-Resolution Modeling and Locally Refined Collision Detection for Haptic Interaction

Liu, P.; Georganis, N.; Roth, Gerhard

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

*Multi-Resolution Modeling and Locally Refined Collision Detection for Haptic Interaction**

Liu, P., Georganis, N., and Roth, G.
June 2005

* published at the Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM 2005). June 13-17, 2005. Ottawa, Ontario, Canada.NRC 48246.

Canada

Multi-resolution Modeling and Locally Refined Collision Detection for Haptic Interaction

Peiran Liu, Xiaojun Shen, Nicolas Georganas
University of Ottawa
[peiran, shen, georganas]@discover.uottawa.ca

Gerhard Roth
National Research Council of Canada
Gerhard.Roth@nrc-cnrc.gc.ca

Abstract

The computational cost of a collision detection (CD) algorithm on polygonal surfaces depends highly on the complexity of the models. A novel "locally refined" approach is introduced in this paper for fast CD in haptic rendering applications, e.g. haptic surgery and haptic sculpture simulations. Exact interference detections are performed on proposed locally refined meshes, which are in multi-resolution representation. The meshes are generated using mesh simplification and space partition. A new BVH algorithm called "Active Bounding Tree", or AB-Tree, handling collision queries is introduced. At runtime the meshes are dynamically refined to higher resolution in areas that are most likely to collide with other objects. The algorithms are successfully demonstrated in an interactive haptic environment. Compared to existing CD algorithms on single resolution models, noticeable performance improvement has been observed in terms of the precision of collision queries, frame rate, and memory usage.

1. Introduction

Interactive virtual environment (VE) requires natural and real-time interaction between computer systems and users. Compared to presentation of visual and auditory information, methods for haptic display are not as well developed. Haptic rendering as an augmentation to visual and auditory displays can enrich the perception and understand both of force fields and of world models populated in synthetic environments.

Haptics, term which was derived from the Greek verb "to touch", introduce the sense of touch and force in human-computer interaction. Haptics enable the human operator to manipulate the environment in a natural and effective way, enhance the sensation of "presence", and provide information such as stiffness and texture of objects, which cannot be described completely with visual or audio feedback only. Early work was accomplished over three decades ago for tele-robotics applications. The potential of the technology is significant for interactive virtual reality, tele-presence, tele-medicine and tele-manipulation applications [1,3,4]. The technology has already been explored in contexts as diverse as modeling & animation, geophysical analysis, dentistry training, virtual museums, assembly planning, mine design, surgical simulation, design evaluation, control of scientific instruments, and robotic simulation. However its true potential in these areas has yet to be achieved, and its application to all aspects of dexterous training, for example, is almost completely untapped. Haptic interaction of virtual environments involves augmentation of a client station and may be viewed as a simple integration of conventional VEs with haptic displays. The application family is typically implemented upon non- or soft- real time operating systems. By contrast, tele-haptic interaction imposes more stringent requirements and, like tele-robotics, demands hard real time guarantees [1].

Multiple tasks, such as haptic sensing/actuation, visual updates must be accomplished in a synchronized manner in haptic applications. It becomes commonplace to separate tasks into computational threads or processes, to accommodate different update rates, distribute computation load, and optimize computation. Conventionally, multithreading and multiprocessing software architectures are applied to develop effective multimodal VEs and the optimal usage of the CPU capabilities [6]. However, an important but little discussed consequence of the

Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)
1550-6185/05 $20.00 © 2005 **IEEE**

conventional architectures is that it makes the operating system an inherent component of the applications, with operating system scheduling algorithms limiting the application's quality of service. The application may request a theoretical rate of force display but it is the scheduler that determines the actual rate. This scheduler is itself a complex algorithm, particularly when considered in terms of its interactions with the other services provided by the operating system [5].

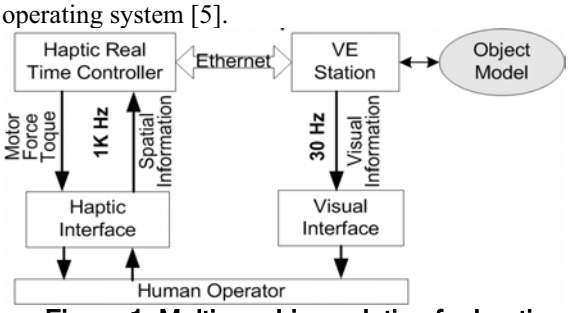

Figure 1. Multi-machine solution for haptic applications

A multi-machine solution for haptic application was addressed in [1,2] as shown in Figure 1. The multi-machine architecture is comprised of three parts: haptic device, Haptic Real Time Controller (HRTC) and Virtual Environment (VE) graphics station. HRTC communicates with its VE station through a local Ethernet connection. HRTC relies on hard real time operating systems (eg. QNX Neutrino, VxWorks or Windows CE) to guarantee the stability of the control loop.

The separation of functionalities of haptic and graphic rendering makes the proposed architecture easier to extend to existing applications. Unlike conventional multithreading or multiprocessing approaches for haptics, this multi-machine model solution applies a hard real-time operating system for haptic control, while applying a mainstream OS such as Win2K or WinXP for the application and graphics. It also provides the potential for tele-haptic applications to switch between multiple protocols, one for large-scale distributed simulations and one adapted to collaboration when several users meet and need to perform a collaborative task, for example, and requires an architecture that supports those different protocols.

Two major tasks in haptic interaction paradigms are *collision detection* (CD) and *collision response*. Collision detection is to detect collisions between the end point of a generic probe and the objects in a scene, while collision response is to respond to the detection of collision in terms of how the forces reflected to the user are computed. Studies of human tactile perception of contact information have shown that a desired force update rate is preferably 1 kHz.

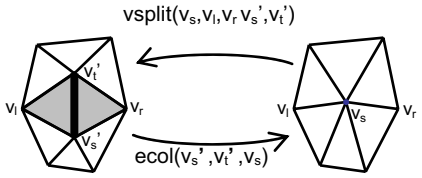

Figure 2. Vertex split and edge collapse

Although there is a huge volume of literature in the area of collision detection, many proposed algorithms targeted graphics applications which require a relatively low collision query rate (desirably 30Hz). Existing collision detection algorithms proposed so far only support static level-of-details (LOD) meshes. The running time of the algorithms depends on the complexity of input models and the output collision configuration. Therefore, setting the resolution of the meshes for collision detection in building a virtual environment becomes an engineering trade off between speed and accuracy. In graphic applications, rendering complex models with millions of polygons at interactive rate can be handled by modern GPU. However, an efficient algorithm has not emerged for creating an interactive force display on the complex models.

Based on recent research in mesh simplification and bounding volume hierarchy (BVH) CD, we propose here our locally refined CD approach. It is illustrated in this paper that it is possible to increase the accuracy of collision detection without sacrificing speed, using a space partitioned multi-resolution mesh representation and a BVH that we call an Active Bounding Tree, or AB-Tree. The space partition technique makes it possible to locally refine some parts of a mesh that are most likely to collide with other models in the near future. Collided parts of a mesh can always achieve the finest resolution. The other parts remain in coarse resolution. In such a way, the input size of the CD is decreased and the hard real-time performance requirement for haptic interaction is achieved. Highly detailed geometric models are becoming necessary to fulfill a growing expectation for realism in haptic rendering. However, the huge data volume exceeds the memory size of current haptic rendering hardware. The progressive techniques can bridge the gap between hardware capability and complexity of the geometric models by selectively loading the meshes at runtime from external memory. The local memory only keeps currently refined mesh data that are necessary for CD plus some pre-fetched data that are needed for CD in the future.

Main contribution: A real-time collision detection framework for haptic interaction between multiple probes and complex object models is proposed. The

high update rate of force display and the limited memory capacity of haptic rendering hardware are two of the most challenging issues we resolved. A new "Space Partitioned Multi-resolution (SPM)" mesh representation is introduced. The meshes can be progressively transmitted on demand and locally refined at runtime. A new BVH CD algorithm, AB-Tree, is introduced which performs output-sensitive CD for dynamically refined meshes at a cost similar to that of the existing CD algorithms on static LOD meshes. The extra cost, mesh refinement, is as low as a small constant when the objects do not move swiftly very often. The algorithm allows us to bound the input size of the problem, thus achieving the desired collision query performance for force display.

Organization: The rest of the paper is organized as follows. We start with a brief survey of collision detection and view-dependent meshes in section 2. In section 3, we introduce the proposed CD framework for haptics. The AB-Tree algorithm is described in section 4. Section 5 presents the performance of our implementation. Summary and discussion are given in section 6.

2. Related work

2.1. Collision detection

An accurate and fast collision detection algorithm is considered to be one of the major bottlenecks in building interactive and realistic haptic environments. Many important methods have been developed for rigid models. Recent surveys of these methods can be found in [7,9]. Algorithms with the current best run time for convex polytope collision queries take linear time. If the objects are not moving swiftly, the best runtime can be roughly constant. The algorithms for collision queries between general non-convex polygonal models are dominated by the hierarchical bounding volume (BVH) strategy. The time cost is mainly determined by the complexity of input model, the choice of bounding volume (BV), and the contact configuration of two models. Sphere-trees [8], oriented bounding boxes (OBB) trees [10], axis-aligned bounding boxes (AABB) trees [11], discrete orientation polytopes (k-dops) trees [12], SSV-trees [13], and CLOD [14] are examples. BVHs can be based on either spatial proximity between features of a model, mesh topology, or inter-surface proximity [15]. In cluttered environments, OBB trees perform better than AABB trees and sphere trees due to the tight fitting bounding boxes. Although AABBs cannot fit some primitives like long-thin oriented polygons tightly, a BVH based on AABBs performs overlap test faster compare with OBB trees. Furthermore, AABB trees need less memory and are easy to update after deformation. In time-critical collision detection, the output precision is allowed to be gracefully adapted to the computing time available [8,16]. However, spatial inaccuracy is inherent due to the lack of exact collision queries on primitives. This is especially important when contact normal and contact points are required to compute a plausible collision response.

2.2. View-dependent multi-resolution meshes

Detailed models result in large storage space, expensive transmission cost, and slow geometry manipulation. To address these issues, several mesh simplification techniques have been proposed. Previous mesh simplification works fall into two categories: multiple static LODs and single LOD with multi-resolution. The idea of the multi-resolution mesh or continuous LOD [17,18,19,20,21] is to create a data structure that can be employed to dynamically produce a mesh with any desired resolutions lying between the highest and the lowest number of polygons from the original mesh. Most works for general polygonal models are related to progressive meshes. Progressive mesh is proposed in [17] for accelerating graphics rendering. A triangle mesh can be represented at multiple LODs by performing a series of refinement operations. The operations include vertex split and edge collapse as shown in figure 2. A triangle mesh is encoded as a base mesh plus a sequence of n vertex split records, ($M_0, \{vsplit_0, \ldots, vsplit_{n-1}\}$). Recent works have been focused on view-dependent simplification, which take into account viewing parameters in mesh simplification to speed-up graphics rendering further. In [22] a view-dependent simplification algorithm for progressive meshes in introduced which use screen space projection and viewing orientation to guide the runtime simplification. Luebke and Erikson [23] define a tight octree called vertex tree over the given model to generate hierarchical view-dependent simplifications. Taubin et al [24] demonstrate a surface partition scheme for progressive encoding of surface. Schilling and Klein [25] have introduced a texture dependent refinement algorithm.

A few research works have been focused on using multi-resolution representations for haptic rendering. Pai and Reissel [26] investigate the use of multi-resolution image curves for 2D haptic interaction. El-Sana and Varsheny [27] introduced a multi-resolution hierarchy. A detailed mesh is used for regions around probe pointer and a coarser mesh is used elsewhere. Otaduy and Lin [14] developed an algorithm to

construct a multi-resolution BVH in convex pieces for pair-wise collision queries. This algorithm does not provide simplified meshes that consistently approximate the perceptually detailed realistic models for rendering. The mesh representation does not support progressive transmission. Therefore the algorithm is not appropriate for haptic running environments with limited memory space.

3. Collision detection framework for haptic interaction

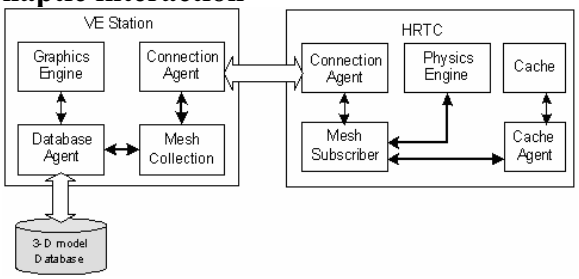

Figure 3. CD framework from the perspective of multi-machine model for haptic applications

In the aforementioned multi-machine haptic model, the VE station is free to handle other tasks of the application, such as graphics rendering, database access, and networking. Another significant benefit is memory saving. The host memory can be used as an external memory for HRTC to access the object models. The goal of this research is to design a CD framework based on such a multi-machine solution. The proposed framework is composed of two parts: the VE station and the HRTC. The HRTC is responsible for mesh refinement, collision prediction, and mesh subscription from the VE station, cache management, and collision detection, as shown in figure 3. The host is responsible for handling the subscriptions from the HRTC, loading meshes to host memory, and progressive transmission of the data. The proposed framework mainly addresses the issues of performing real-time collision detection on complex models using limited memory space.

As mentioned in section 1, the runtime performance of a CD algorithm is directly affected by the complexity of the input models. We developed a local refinement algorithm on progressive mesh to reduce the combinatorial complexity of the input models. The selection of refinement region on the mesh is dependent on the local neighborhood of potential collision. The CD algorithm has two phases, the preprocessing phase and the runtime phase as follows.

Procedure **PreprocessingPhase**
1 Subscribe coarsest meshes from the host
2 Load received mesh models to vertex forest hierarchy
3 Build AB-Tree BVHs on vertex forest hierarchies

Procedure **RuntimePhase**
1 **FOR** each frame
2 Apply refinement prediction to the current meshes
3 Load mesh data from local cache
3 Subscribe and receive mesh data from host
4 Update vertex forest hierarchy and refine the meshes
5 AB-Tree refitting
6 Pair wise interference detection
7 **ENDFOR**

In the preprocessing phase, the volume of transmitted mesh data is relatively small to that of the whole mesh. The structures of AB-Trees are encoded in the SPM meshes which saves the time for BVH construction. Therefore, the cost of running the first phase is neglected. In the runtime phase, the algorithm estimates the time it can spend per frame on collision detection, which is determined by the application's performance goals and the set of activities it performs at each frame. Initially AB-Trees are built on the coarsest meshes. Then some mesh primitives are locally refined to lower or higher resolution, based on the configuration of the objects in space. Then, the AB-Tree BVHs of the models are refitted. Finally, pair wise collision queries are performed on the models. The experimental results introduced in section 6 demonstrate significant performance improvement over existing algorithms for static LOD meshes.

3.1. Space partitioned multi-resolution (SPM) modeling

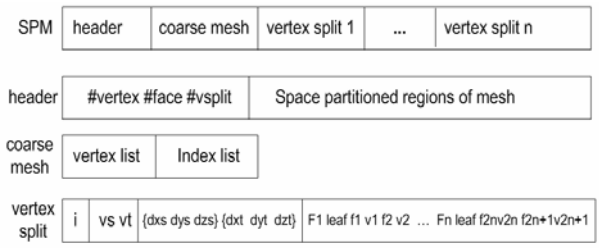

Figure 4. Space partitioned multi-resolution mesh

The proposed SPM modeling method takes advantage of the Progressive Mesh (PM) and the Quadric Error Metrics [28] which is a method we use to efficiently generate simplified versions of traditional triangle meshes in arbitrary topology. A SPM mesh is generated offline based on the PM format, and the vertex split and vertex pair contraction operations. The BV of each model is evenly partitioned to regular regions. Those regions containing mesh primitives are labeled. The format of a SPM mesh for streaming is given in figure 4. Statistical information of the mesh and the labeled

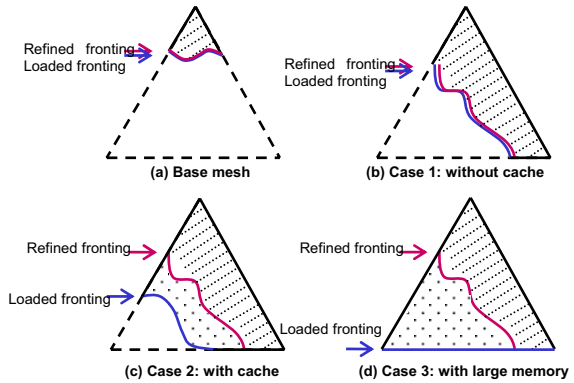

Figure 5. Vertex forest hierarchy

space partition regions are encoded in the header. The structure of the AB-Tree BVH built upon the finest mesh is encoded in vertex split records.

At runtime, a vertex forest hierarchy is created for each loaded mesh model during the preprocessing stage. Every node of the hierarchy stores a vertex split information. The hierarchical data structure records the history of vertex split and edge collapse of a mesh at an instant resolution, which enables fast mesh split and merge. As shown in figure 5, the vertex forest hierarchy maintains two boundaries, *refined fronting* and *loaded fronting*. The refined fronting defines a sub vertex forest in which the vertex split operations stored in each node are performed. The loaded fronting defines another sub vertex forest which comprises a set of vertex split nodes that has been subscribed and loaded in local cache. In preprocessing phase, the two boundaries are overlapped (figure 5a). In runtime phase, the two boundaries are kept overlapping when cache is not available (figure 5b). The space between the two boundaries is increased when more cache is available (figure 5c). Ideally, the cache space is comparable with host memory. In such case, the mesh data are not necessarily subscribed for more than once (figure 5d). When the cache is small, the pre-fetching of the data from host memory to the cache is more frequent which obviously wastes host memory and increases waiting time at the HRTC. By utilizing temporal and spatial coherence and simple motion prediction techniques, the pre-fetching of mesh data can be significantly reduced even when the cache space is relatively small.

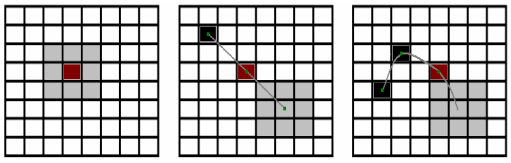

Figure 6. Contact region prediction Left: local neighbors of contact region Middle: linear extrapolation Right: polynomial extrapolation.

3.2. Prediction

Temporal and spatial coherence: Frames in an interactive viewing session typically exhibit only incremental shifts in contact local neighbor, so the number of potential contact regions remains roughly small and constant. Linear and quadratic extrapolation is considered to be at the heart of the best techniques for spatial motion prediction which requires the recording of the contact regions in previous frames. A simpler solution is to take the local neighbors on contact regions in the current frame as the contact regions for the next frame (figure 6). The red areas are contact region in current frame. The black areas are contact regions in previous frames. The grey areas are predicted for next frame.

Distance query: Calculating the distance between probe points and the labeled regions of mesh models can accelerate collision detection. Mesh data contained in the regions which are in a distance smaller than a threshold to probe points are pre-fetched. This prediction can be applied when no contact region is detected in the current frame.

The algorithm for generating the SPM meshes and the vertex forest hierarchies built upon the meshes is general and applicable to other progressive mesh representations.

3.3. Mesh refinement

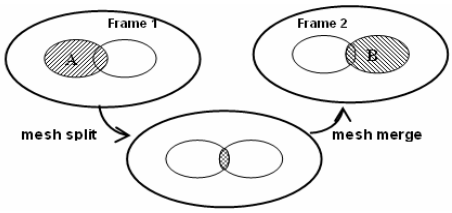

Figure 7. Two-phase local refinement

The frame-to-frame local refinement of a SPM mesh is illustrated in figure 7. Let A and B be two sets of regions. In the regions of A, mesh primitives are refined to full resolution in frame 1. In the regions of B, mesh primitives are refined to full resolution in frame 2. A two-phase (mesh split and mesh merge) operation is required to refine the mesh according to changing contact configuration. In phase one, the edge collapse operations clustered in the regions A/B are collected from the vertex forest hierarchy. Then operations are performed in order on the mesh. In phase two, the vertex split operations clustered in the region B/A are collected from the hierarchy. Then the operations are performed in order on the mesh.

4. Active bounding trees

This section introduces the AB-Tree, a new BVH algorithm for collision query on dynamically refined multi-resolution meshes. Without loss of generality, we consider AABB Trees constructed on polygonal models and briefly discuss how this could be extended to other BVs.

4.1. AB-Tree construction

An AABB tree is introduced in [11] to implement a BVH. It provides a fast way to perform exact collision detection between complex models. In general, it is a binary tree structure. AABB-Trees allow the cost of refitting an AABB tree in an AABB tree to be independent of the number of nodes in the tree.

An AB-Tree T augments an AABB tree with additional information stored in each node and a primitive index list as illustrated in Figure 8. Every node x in T has three additional fields, a link to an element of the index list, $index[x]$; a link to the parent of x, $parent[x]$; and a status of the node x, $status[x]$ where

$status[x] \in \{active, inactive, deformed\}$

An element f in the index list has two fields, a link to a leaf node in the tree, $leaf[f]$, and a link to a primitive in the primitive list of the multi-resolution data structure, $primitive[f]$. Every leaf in the AB-Tree is pointed from exactly one element of the index list.

Since a multi-resolution mesh is refined at runtime, its geometry and topology are changed dynamically. A mesh primitive may be inserted, removed, or deformed. Therefore, in order to perform exact collision detection on the mesh, a fast algorithm is required to refit the BVH representation of the mesh. First, locating a set of leaves and all of their ancestors in the BVH needs to be fast, given that the primitives bounded by the leaves are known. In an AABB tree, this may require searching several paths top-down due to the possible overlaps between the BVs of the sibling nodes. However, in an AB-tree, the index of a primitive can be used as a key to the index list so that the leaf which bounds the primitive can be located in a constant time cost. The parent field can then be used to find all the ancestors of a leaf in a bottom-up tracing. Compared with the AABB tree, it is clear that the AB-tree has better performance. Second, the active primitives of a refined mesh and their BVs need to be maintained efficiently such that the complexity of collision queries performed on the BVH is proportional to the mesh complexity. In other words, when the mesh is refined coarser, the collision queries run faster. The *status* field works for the second requirement.

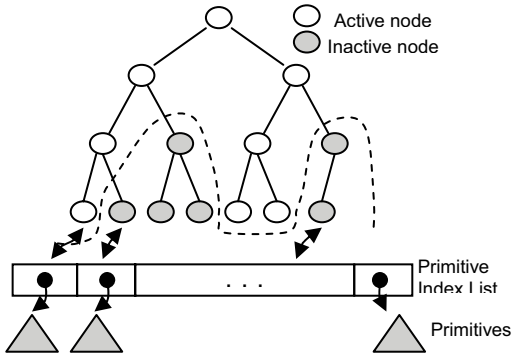

Figure 8. AB-Tree data structure

4.2. Fast BVH refitting

One assumption of the proposed BVH algorithm is that a mesh is refined gradually instead of been refined in a drastic way. This means only a few primitives of the original mesh are affected in each refinement. Therefore only a small number of BVs in a BVH need to be refitted for collision detection in each frame. To quickly locate these BVs, a bottom–up approach can be much faster than a top-down approach, given that the affected primitives are known from the mesh refinement, as discussed in the last section. The index list and the parent pointer in the tree are designed for this purpose. Once a node is located, the *status* field is used to identify those BVs to be refitted. *deformed* means that the BV needs to be refitted; *inactive* means that the BV is temporarily removed; *active* means that there is no change. Initially all nodes in the tree are set to *inactive*. All those *deformed* nodes form a small subtree. When a top-down traverse is performed on the subtree, the refitting is applied to each *deformed* BV. Comparing with the AABB tree which requires a traverse on the whole tree, our active tree algorithm is much faster. An AB-tree maintains a set of dynamically changed BVs of a BVH.

Lemma 1

For a binary tree *Tc* which has n leaves and height $O(\log n)$, marking k randomly chosen leaves and all of their ancestors requires marking $O(k\log n - k\log k + 2k)$ nodes in worst-case.

Proof: Lemma1 is proved in our previous work [29].

Theorem

Let T_a be an AB-tree built upon a SPM mesh P. F is the index list of T_a. A BVH refitting operation takes $O(k(\log n - \log k + 2))$ time where n is the number of leaves in T_a, k is the number of primitives to be deformed in, removed from, or inserted into the mesh P at an instant resolution. Furthermore, if k is set to be

in a range [1,K], where K is a constant and $K << n$, then the call takes $O(\log n)$ time.

Proof: The theorem can be easily proved by lemma 1.

Table 1. Parameter settings for models

Models	Sphere	Cow	Bunny
# faces in M_n (Original Mesh)	4096	5804	37576
# vertices in M_n(Original Mesh)	2050	2904	20000
# faces in M_0 (Base Mesh)	50	704	500
# vertices in M_0 (Base Mesh)	27	355	1406
# vertex split records	2023	2549	18594
AB-Tree height	12	12	16

5. Runtime performance

We have successfully applied our approach to interference detection on the benchmark models given in table 1. Two models are shown in Figure 9. Experimental results are given in table 2. The demonstrations have been run on dual Pentium4 2.8GHz processor PCs with 510MB of RAM and Windows XP OS to simulate a multi-machine haptic environment. The real-time graphic rendering is achieved with NVIDIA® GeForce™ FX5200 Graphics Cards. Our implementation uses C++ and an OpenGL library for physics simulation and graphics rendering.

In the HRTC, upon receiving mesh refinement data, time for collision detection can be expressed as

$$T = T_R + T_B + T_Q + T_L$$

, where T_R represents the time for mesh refinement, T_B represents the time for BVH refitting, and T_Q represents the time for collision queries on the BVH. T_L is the time for mesh loading. Assuming that the collision query frame rate is fixed, T_L is proportional to the number of vertex split records loaded per frame. T_L increases when the cache space is decreased. Table 2 reflects the time for T_R, T_B and T_Q. When contact location does not change drastically, the time cost for mesh refinement is near a small constant. With this assumption, T_B is observed to increase in the order of the logarithm of the size of the full meshes. This observation is consistent with the proven theorem introduced in section 4. In terms of T_Q, we find out that the time for collision queries in a low resolution mesh is only 0.5% less than the time cost in a locally high resolution mesh, whereas, the cost for the finest resolution mesh is 3 times more than the cost for refined meshes. In terms of memory usage, only the refined regional mesh data are kept in cache. Mesh reloading takes place once in a few seconds. This memory saving strategy makes complex models easy to handle in hapic applications.

Table 2. Performance statistics for bunny and probe models

Time	Original mesh in static LOD	Base mesh in static LOD	SPM with local refinement
T_Q	5.7ms	2.75ms	2.87ms
T_B	n/a	n/a	1.8ms
T_R	n/a	n/a	5.4ms

6. Summary and Discussion

In conclusion, the performance of the proposed CD algorithm is slightly affected by the swift movement of the objects. However, we propose a divide-and-conquer algorithm to successfully break the constraint of large input size of complex models. The algorithm partitions large models into separate regions and selectively performs collision queries on them. One heavy computing task is divided into many subtasks. From the end users' point of view, they can start running a haptic application without knowing the whole geometric environment. A realistic interactive force display is achieved smoothly and instantly.

We believe that locally refined CD is a fresh starting point for future work on multi-machine haptic interaction. However, current implementation is limited to complex polygonal models without swift movement. A possible extension is to apply the algorithm to other type of multi-resolution representations. Optimizing intelligent motion prediction to further reduce the time for transmission of data is a promising topic to be further studied.

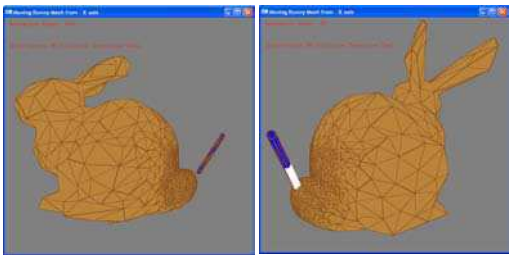

Figure 9a. Local refine SPM mesh Left: contact region prediction; right: interference detection.

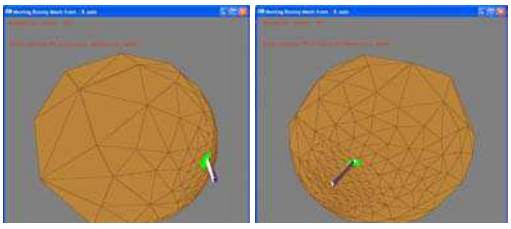

Figure 9b. Frame-to-frame refinement

7. References

[1] X. Shen, J. Zhou, A. El Saddik, and N. D. Georganas, “Architecture and Evaluation of Tele-Haptic Environments”, Proc. 8th IEEE International Symposium on Distributed Simulation and Real Time Applications (IEEE DS-RT 2004), October 2004, Budapest, Hungary

[2] X. Shen, F. Bogsanyi, L. Ni, and N. D. Georganas, “A Heterogeneous Scalable Architecture for Collaborative Haptics Environments," 2nd IEEE Workshop on Haptic, Audio and Visual Environments and their Applications - HAVE, September 2003

[3] J. Zhou, X. Shen and N. D. Georganas, “Haptic Tele-Surgery Simulation”,Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2004.

[4] N. R. El-Far, X. Shen, and N.D. Georganas, “Applying Unison, a Generic Framework for Hapto-Visual Application Development, to an E-Commerce Application”, Proc. IEEE Workshop on Haptic Audio Visual Environments and their Applications, Ottawa, Canada, October 2004.

[5] A. Kirkpatrick and J. Sze, Operating-System Induced Jitter in Force Display Computations, 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS 2004), 27-28 March 2004, Chicago, IL, USA

[6] Ho, C., Basdogan, C., Srinivasan, M.A., “An Efficient Haptic Rendering Technique for Displaying 3D Polyhedral Objects and Their Surface Details in Virtual Environments (PDF)”, October’99 Vol. 8, No. 5, pp. 477-491, Presence: Teleoperators and Virtual Environments, MIT Press.

[7] P. Jiménez, F. Thomas, and C. Torras, *Collision Detection: A Survey.* Computers and Graphics, Vol. 25, No. 2, 2001, pp.269-285.

[8] P.M. Hubbard, Approximating polyhedra with spheres for time-critical collision detection. ACM Transactions on Graphics Vol. 15, Issue 3, 1996, pp.179-210.

[9] M. Lin and S. Gottschalk, Collision Detection between Geometric Models: A Survey. In Proc. of IMA Conference on Mathematics of Surfaces, 1998, pp. 37-56.

[10] S. Gottschalk, M. Lin and D. Manocha, OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. In Proc. of ACM SIGGRAPH'96, pp. 171-180, 1996.

[11] G. van den Bergen, Efficient Collision Detection of Complex Deformable Models using AABB Trees. Journal of Graphics Tools, 2(4), 1997, pp.1-13.

[12] J. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Trans. Visualization Comput. Graph., vol. 4, no. 1, pp. 21-37, 1998.

[13] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Tech. Rep. TR99-018, Dept. of Comput. Sci., Univ. North Carolina, 1999.

[14] M. A. Otaduy and M. C. Lin, CLODs: Dual Hierarchies for Multiresolution Collision Detection. In Proc. of Eurographics Symposium on Geometry Processing, Aachen, Germany, 2003, pp. 94-101.

[15] R. Bridson, R.P. Fedkiw, and J. Anderson, Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. ACM Transactions on Graphics, Vol. 21. No.3, pp. 594-603.

[16] G. Bradshaw and C. O’Sullivan, Adaptive Medial-Axis Approximation for Sphere-Tree Construction. ACM Transactions on Graphics, Vol 23, No. 1, 2004, pp 1-26.

[17] H. Hoppe, Progressive Meshes. In Proc.of SIGGRAPH'96, 1996, pp. 99-108.

[18] M. Gross, O. Staadt, and R. Gatti, Efficient Triangular Surface Approximations using Wavelets and Quadtree Structures. IEEE Trans. on Visual and Computer Graphics, 2(2), 1996, pp.130-144.

[19] J. Rossignac and P. Borrel, Multi-resolution 3D Approximation for Rendering Complex Scenes. In Geometric Modeling in Computer Graphics. Springer Verlag, 1993, pp.455-465.

[20] P. Lindstrom, D. Koller, W. Ribarsky, L. Hughes, N. Faust, and G.Turner, Real-time Continuous Level of Detail Rendering of height fields. In Proc. of ACM SIGGRAPH’96, Aug. 1996, pp. 109-118.

[21] J. Xia, J.El-Sana, and A. Varshney, Adaptive Real-time Level-of-Detail Based Rendering for Polygonal Models. IEEE Transactions on Visualization and Computer Graphics, Vol.3, No.2, June 1997, pp. 171-183.

[22] H.Hoppe. View-Dependent Refinement of Progressive Mesh, In Proc. of *ACM* SIGGRAPH’97, August 1997, pp.189-198.

[23] D. Luebke and C. Erikson, View-Dependent Simplification of Arbitrary Polygonal Environments, In Proc. of SIGGRAPH’ 97, August 1997, pp. 199-208.

[24] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus, *Progressive Forest Split Compression*, In Proc. of SIGGRAPH'98, 1998.

[25] A. Schilling and R. Klein. Texture-dependent Refinement for Multi-resolution Models. In Computer Graphics International, June 1998.

[26] D.K. Pai and L.M. Reissel, Haptic Interaction with Multiresolution Image Curves. Computer and Graphics 21, 1997, pp. 405-411.

[27] J. El-Sana and A. Varshiney, Continuously-adaptive Haptic Rendering. Virtual Environments 2000, pp. 135-144.

[28] M. Garland and Paul S. Heckbert, *Surface Simplification using Quadric Error Metrics*. In Proc. of SIGGRAPH '97, 1997, pp. 209-216.

[29] P. Liu, N. Georganas, and G. Roth, Handling Rapid Interference Detection of Progressive Meshes Using Active Bounding Trees. Journal of Graphics Tools, submitted.