Locally Refined Collision Detection of Large Scale Complex Polygonal Meshes in Distributed Virtual Environments

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

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

Abstract

This paper presents a new *locally refined collision detection* approach for large scale complex meshes in distributed virtual environments (DVEs) where exact and interactive interference detection is required. Transmitting models with millions of polygons is time consuming in comparison with transmitting simple models. Even if the models are transmitted in progressive manner, the earlier received models are in low level-of-detail (LOD) because the models are refined globally. Increasing the accuracy of collision detection (CD) at the client still takes time because increasing the LOD of the model is a slow process. The new approach is composed of an AB-tree collision query algorithm and a new mesh refinement algorithm on a space partitioned mesh (SPM) representation. It deals with this problem by selectively refining the models at certain areas that are predicted to collide with other objects and transmitting the refined parts instead of the entire model from server to client. The accuracy of CD is increased quickly at the predicted contact areas. An interactive rate is guaranteed by reducing network response time and dynamically adjusting the complexity and the space cost of the collision query algorithm.

Keywords: Multi-resolution mesh, bounding volume hierarchy, collision detection, distributed virtual environment.

1. Introduction

A distributed virtual environment (DVE) is a network-based multi-user VR system where participants navigate in synthetic space and see, meet and interact with other users and computers. Two main approaches have been proposed for DVEs to distribute virtual objects from servers to clients. *Complete replication* is an approach to distribute geometry data to the clients before the simulations are started. This approach assumes the use of high-speed networks to transmit the usually large volume of data. *On-demand transmission* is another approach to distribute the data to the clients at runtime. This approach requires the transmission of only the visible region of the virtual environment to the clients, which reduces startup time and optimizes network usage. The visible objects need to be retrieved from the server in advance so that they are available whenever the clients need them. However, there is a problem of maintaining an interactive rate at the clients. It has been solved by scheduling methods [1] and by pre-fetching and caching methods [2]. These solutions only focused on providing continued viewing service of the virtual environments. However, dynamic simulation, especially the issue of interactive and accurate collision detection over the distributed environment, is not tackled in the abovementioned works.

In this paper, we present a new approach, "locally refined collision detection", based on a form of bounding volume hierarchy. The new approach applies to large scale complex models, such as terrain models, where the collision typically occurs in a small area of the entire mesh surface. The main idea of the approach is to refine the meshes locally where collisions are most likely to occur. In a flight simulation, for instance, as an aircraft flies over a terrain mesh, the regions near the aircraft which are at high risk of collision would be selected to refine to higher resolutions. The major benefit of this approach is that the other regions of the mesh may not be required by the collision detection (CD) algorithm. Therefore for complex models only a small portion of the data needs to be loaded from server to client. This can significantly reduce network traffic, offload the workload of the server, and accelerate simulation at the client. In addition, the accuracy of CD is increased when collisions occur in the selected regions of the meshes. When relative position and orientation among objects change, the selected regions are changed dynamically. The un-selected regions only need to be refined to the coarsest resolution. Thus, when refinement data can not be received by the client in time, CD is still supported although in a low accuracy.

Contributions: A framework for exact and interactive CD on large scale complex model in distributed virtual environments (DVEs) is presented. A multi-resolution mesh representation SPM is defined to fulfill the requirement of local mesh refinement. A refinement criteria involving time and space coherence is presented. An efficient mesh refinement algorithm on the SPMs is presented. Its correctness is proved. An effective BVH CD algorithm ("AB-tree") is applied to the proposed multi-resolution progress meshes.

2. Related Work

In this section, we provide a brief review of multi-resolution meshes and collision detection methods. Then we justify the importance of our work and how it differs and improves on the work of others.

2.1 Multi-resolution Meshes

Multi-resolution mesh simplification creates 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 at run-time. Some multi-resolution mesh simplification algorithms refine or simplify meshes globally based on the distance to the view point and projected area on the screen. They are called *continuous LOD* [3, 4]. Continuous LOD allows a geometric model to be efficiently delivered over a network in a progressive manner. Compared to continuous LOD representations, view-dependent LOD representations have better granularity. They allocate polygons where they are most needed, within as well as among objects, therefore enabling even better fidelity in graphics rendering. They enable drastic simplification of very large objects, such as the stadium models and terrain models. Recent works on view-dependent simplification take into account viewing parameters in mesh simplification to speed-up graphics rendering further [5, 6, 7, 8].

2.2 Collision Detection

Collision detection is considered to be one of the major bottlenecks in building interactive and realistic virtual environment. Some recent surveys of the works for CD can be found in [9, 10]. Most of CD algorithms support static LOD, which means that the mesh geometry and topology are static at runtime. In DVEs, they are predetermined by the lowest available network bandwidth and the accuracy of CD required by the application. A few recent works have been focused on using multi-resolution representations in CD [11, 12, 13]. The above mentioned algorithms do not support progressive transmission of the models over the network. No good real-time collision detection systems are known for distributed and interactive virtual environments, especially those composed of large scale complex models. We are seeking to overcome the shortcoming of the existing strategies and propose new strategies to speed up collision detections in the context of distributed virtual environment.

3. Locally Refined Collision Detection in DVEs

A collision detection progressive mesh (CDPM) representation and a globally refined collision detection approach applying to the meshes in DVEs are introduced in [16]. One problem of the approach is that collision queries only apply to globally refined meshes. The resolutions of the meshes cannot be adjusted locally in specified regions on the surface model. The complexity of a BVH collision query algorithm is mainly determined by the number of BV overlap tests while the accuracy of a collision detection algorithm is determined by the resolution of the meshes at contact location. The proposed *local refinement collision detection* is a deviation of the CDPM CD approach which supports fast and accurate CD on large scale complex models in distributed environments. Vertex split records for local refinement of specified areas of the mesh are subscribed by the client and collected and sent by the server. Only those parts of the mesh that are most likely to collide with other objects are refined to full resolution at runtime. This means that the input size of the collision detection process does not changed significantly when the meshes are locally refined. When a comparable accuracy is achieved on the same large model, the locally refined collision detection approach runs faster than the globally refined approach.

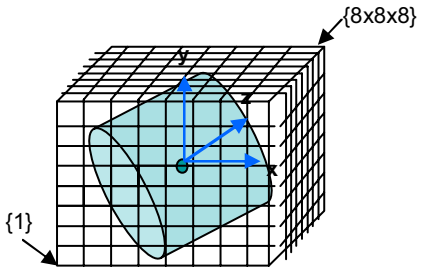

Figure 1 Illustration of space partitioning a polygonal model into 8x8x8 3D grids

3.1 Collision Detection Framework

The framework allows fast and exact interference detection that adapts to local surface mesh refinement and progressive transmission in distributed virtual environments. The framework consists of two parts: server and client. The server is responsible for transmitting base meshes, collecting and transmitting legal vertex split records for local refinement upon requests from the clients. The client is responsible for determining when and where to do local refinement based on a region selection function, sending refinement parameters (indices of certain regions in space that are occupied by the meshes) to the server, receiving vertex split records from the server, performing refinement, building and refitting BVHs of meshes, performing collision queries. The CD algorithm at a client has two phases, preprocessing phase and runtime phase. In the preprocessing phase, the size of the transmitted mesh data is relatively small compared to that of the entire mesh. The structure of AB-Trees for collision query is encoded in the SPMs which saves the time for BVH construction. Therefore, the cost of running the first phase is negligible. In the runtime phase, the algorithm estimates the time it spends 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 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.

3.2 Collision Prediction

SPM models are generated through a space partition process. An object in a scene is fitted by a bounding box. The 3D space of the

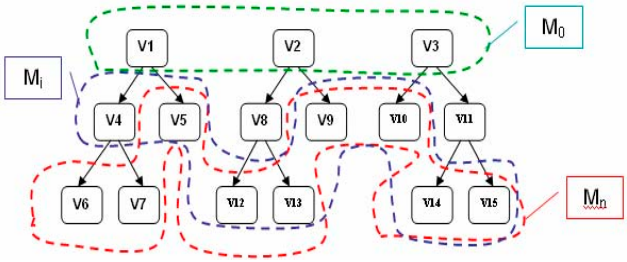

Figure 2 Illustration of a base mesh M_0, a full mesh M_n and a locally refined mesh M_i for SPM. M_i is generated by a sequence of VSPs in a selected order

bounding box is partitioned evenly into smaller indexed boxes. A refinement criterion is required to adapt mesh refinement when relative configurations among objects are changed. As a result, more and more regions indices are reported at runtime, and more parts of the meshes are refined.

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.

3.3 Space Partitioned Meshes and Selective Refinement Algorithm

The proposed SPM modeling method uses the Progressive Mesh [17] and the Quadric Error Metrics (QEM) [14]. The QEM is a method we use to efficiently generate SPM from traditional triangle meshes in arbitrary topology by performing two operations, vertex split (VSP) and its reverse, edge collapse (ECOL). A vertex split operation splits one vertex v_s to two new vertices v_u and v_t, and adds at most two new faces f_l and f_r. SPMs represent manifold triangle meshes. A SPM can be streamed over network by progressively transmitting VSPs in sequence. The format of the SPM is similar to CDPM. The major difference is the index list of space partitioned regions in the SPM header. In order to do selective mesh refinement, an AABB BV is calculated for the original mesh. The BV is evenly space partitioned to 3D grids as illustrated in Figure 1. Each grid is indexed based on the coordinates of the partitioning planes along the axes of the BV. The index list helps the server to collect VSP records that are used by the client to locally refine selected regions on the current mesh.

At the client the SPM multi-resolution data structure is composed of a *vertex hierarchy* and a *sequence of performed VSPs*. The vertex hierarchy is a forest as illustrated in Figure 2. The hierarchy not only contains information about the geometry and topology of the model in a continuous LOD but also records the history of vertex split and edge collapse operations on the multi-resolution mesh, which enables fast mesh split and merge. The leaf nodes are

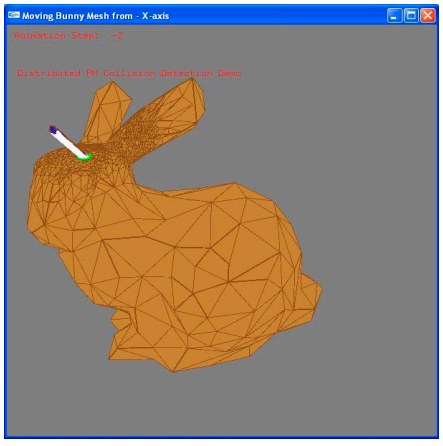

Figure 3 Illustration of locally refined collision detection

extended dynamically upon receiving vertex split records. A leaf node can extended to have two children to split the leaf's represented vertex into two vertices and generate new faces in the mesh. On the other hand, the extended nodes can be folded into their parent to collapse the split two vertices to one and removed the generated faces from the mesh. The sequence of performed VSPs is designed to record the order the vertices represented by the nodes in the hierarchy are split and to quickly locate the nodes on the hierarchy to be folded or unfolded.

In a distributed environment, tasks at the client include collision prediction, collision region VSPs subscription, receiving VSPs data, SPM multi-resolution refinement, and collision queries [15]. The AB-tree collision query algorithm has the characteristic that it receives all the refined faces in the current frame no matter whether the meshes are refined globally or locally. The fundamental mesh refinement operations associated with the SPM multi-resolution data structure are Local Edge Collapse and Local Vertex Split. Local Vertex Split is performed when the predicted collision regions are different from current collision regions. It collects all required VSPs from the index list of partition regions in the received mesh header. Then for each collected VSP, if the legal conditions introduced in [6] are satisfied, the VSP is performed on the hierarchy. Otherwise, a set of ECOLs and VSPs are collected and performed to make the VSP legal. This operation is a recursive process which performs all necessary VSPs and ECOLs to refine the predicted regions to the highest LOD. It terminates when all vertices in the predicted regions on the mesh become active. Local Edge Collapse is performed when the current collision regions are predicted not to collide in the next frame. This operation is another recursive process which performs all necessary VSPs and ECOLs to refine the current collision regions to the lowest LOD. Both operations update the AB-tree BVH for collision query if the mesh is refined.

Tasks at the server include loading mesh header and base mesh of subscribed models from secondary storage to main memory, receiving collision regions index list from the client, collecting VSPs required for mesh refinement, sending the collected data to the client. The fundamental operation associated with the SPM multi-resolution data structure is Server Collect VSplits which collects VSPs required for mesh refinement. This operation is a simple case of Local Vertext Split which does not consider edge collapse operations because the mesh maintained by the vertex hierarchy at the server is never refined to lower LOD. The proof of correctness of the local refinement algorithm is given in [16].

4. Conclusion

The configuration of benchmark models and simulation results are given in table 1 and 2. Upon receiving mesh refinement data at the client, 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 AB-tree 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. The performance of the proposed locally refined collision detection is slightly affected by the network bandwidth and the movement of

models. However, the initial waiting time at a client is significantly reduced. We propose to segment one heavy computing task into many subtasks. The subtasks in the client are performed in parallel with the tasks in the server through the network. Even when only a small portion of mesh data is received, the accuracy of the collision detection is as high as that can be obtained by performing collision queries on meshes in full resolution. From end users' point of view, they can start running a highly realistic and interactive virtual environment application instantly without waiting for the whole scene to be received.

The AB-tree BVH collision detection, the SPM representation, and the framework for locally refined collision detection in DVEs presented in this paper are successfully integrated. A prototype application has been developed in a distributed setting to demonstrate the feasibility of the framework. A screenshot is provided in Figure 3. The performance is studied both analytically and by measurements. The proposed approach can significantly improve existing collision detection methods in distributed virtual environments especially in those that involves large scale complex models in low bandwidth networks where exact and real-time interference detection in required.

Acknowledgments

I am grateful to the LORNET NSERC Research Network for supporting this research.

References

[1] W. Wong and R. Muntz, *Providing Guaranteed Quality of Service for Interactive Visualization Applications*. In Proc. ACM SIGMETRICS, June 2000.

[2] J. Chim, R.W.H. Lau, H.V. Leong, and A. Si, *Cyber Walk: A Web-based Distributed Virtual Walkthrough Environment*. IEEE Transactions on Multimedia, Vol.5, No.4, Dec. 2003, pp.503-515.

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

[4] N. Molino, Z. Bao, and R. Fedkiw, *A Virtual Node Algorithm for Changing Mesh Topology During Simulation*. ACM Transactions on Graphics (TOG), vol. 23, no.3, August, 2004, pp. 385-392.

[5] 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, Jun. 1997, pp. 171-183.

[6] H. Hoppe, *View-dependent refinement of progressive meshes*. In Proc. ACM SIGGRAPH' 97, August 1997, pp.189-198.

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

[8] C. Touma and C. Gotsman, *Triangle Mesh Compression.* In Proc. Graphics Interface '98, June, 1998, pp. 26-34.

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

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

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

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

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

[14] M. Garland and P.S. Heckbert, *Surface Simplification using Quadric Error Metrics*. In Proc. SIGGRAPH, 1997, pp.209-216.

[15] P. Liu, N.D. Georganas, and G. Roth, *Handling Rapid Interference Detection of Progressive Meshes Using Active Bounding Trees*. Journal of Graphics Tools, accepted for publication in 2005.

[16] P. Liu, *Progressive Transmission and Multi-resolution Collision Detection of Polygonal Meshes in Virtual Environments*. Ph.D thesis, University of Ottawa, 2006.

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

Table 1 Parameter Settings for Models

Models	#faces in M_n (Original Mesh)	#vertices in M_n (Original Mesh)	#faces in M_0 (Base Mesh)	#vertices in M_0 (Base Mesh)	#vertex split records	AB-tree height
Sphere	496	2050	50	27	2023	12
Bunny	37576	20000	500	1406	18594	16

Table 2 Performance Statistics for CD

Model	T_Q for original mesh in static LOD	T_Q for base mesh in static LOD	T_Q for SPM with local refinement	T_B for SPM with local refinement	T_R for SPM with local refinement
Sphere	0.19ms	0.03ms	0.09ms	0.30ms	0.21ms
Bunny	0.42ms	0.05ms	0.35ms	1.80ms	5.40ms