pISSN 1976-913X
eISSN 2092-805X

Enhanced FFD-AABB Collision Algorithm for Deformable Objects

JaeHong Jeon*, Min-Hyung Choi** and Min Hong***

Abstract—Unlike FEM (Finite Element Method), which provides an accurate deformation of soft objects, FFD (Free Form Deformation) based methods have been widely used for a quick and responsive representation of deformable objects in real-time applications such as computer games, animations, or simulations. The FFD-AABB (Free Form Deformation Axis Aligned Bounding Box) algorithm was also suggested to address the collision handling problems between deformable objects at an interactive rate. This paper proposes an enhanced FFD-AABB algorithm to improve the frame rate of simulation by adding the bounding sphere based collision test between 3D deformable objects. We provide a comparative analysis with previous methods and the result of proposed method shows about an 85% performance improvement.

Keywords—FFD-AABB Algorithm, Physically-Based Simulation, Deformable Objects, Collision Detection, Bounding Sphere

1. INTRODUCTION

Due to demands for increasing levels of realism, many interactive computer graphics applications such as computer games, virtual reality, and animation require both reasonably high resolution to represent the geometric details of real objects, and high performance for achieving real-time interactivity. In general, object modeling can be categorized with a physically based modeling and a non-physically based modeling technique. Non-physically based modeling mainly deals with the visual component of the modeling. It is typically focused on the detailed geometric information of objects. On the other hand, a physically based modeling technique emphasizes the dynamic and physical nature of an object behavior under various external influences. It can provide increasing levels of realism for real world deformable 3D [1]. The prevailing issue in deformable object modeling is the conflicting goals between higher physical and dynamic realism on densely meshed objects and achieving faster computation at the same time.

Over the past decades, many researchers have introduced various approaches to model and simulate deformable objects in a practical sense. Although rigid body simulations have been successfully applied in computer graphics and computer games with satisfactory interactivity

※ This work was supported by the National Research Foundation of Korea (NRF) grant, which is funded by the Korean government (MEST) (No. 2011-0014775).
Manuscript received August 14, 2012; accepted October 8, 2012.

Corresponding Author: Min Hong
* Department of Computer Science, Graduate School, Soonchunhyang University, Korea. (jjhong@sch.ac.kr)
** Department of Computer Science and Engineering, University of Colorado Denver, U.S.A. (min.choi@ucdenver.edu)
*** Department of Computer Software Engineering, Soonchunhyang University, Korea. (mhong@sch.ac.kr)

and accuracy, still so many real-world objects cannot be reasonably represented with rigid body approximation. The mass-spring system is one of the fast and simple techniques that has been used to represent deformable objects. In the mass-spring system, deformable objects consist point masses and springs, as Newton's Second Law calculates discrete model and spring forces. FEM (Finite Element Method) and BEM (Boundary Element Method) are continuum mechanics-based approaches that can deliver sophisticated and physically accurate behavior. However, their computational complexity is still quite expensive for real-time applications. Despite the computational complexity issues, these methods have been widely applied to implement the behaviors of clothes [2, 3], human tissues [4, 5], muscles [6, 7], and so on. Alternatively, FFD (Free Form Deformation) based methods also have been used for the modeling and animation of complex and highly meshed 3D objects due to its fast real-time performance, even though it is not a fully physics based approach. In addition, many existing methods such as the mass-spring system [8], FEM [9], and shape-matching deformation [10] were supplemented with FFD-style approaches to improve the realistic and efficient representation of large deformations.

Another serious obstacle in real-time 3D computer animation and games is collision detection and the response process. So far, collision detection and the response for deformable objects have been intensively researched and some significant results have been accomplished [11, 12]. Jimenez et al. [13] provided a comprehensive survey for various 3D collision algorithms. Bounding volume based approaches, such as sphere trees [14] and AABB (Axis Aligned Bounding Box) [15], were introduced to quickly check collisions between 3D objects. James and Pai [16] proposed the Bounded Deformation Tree using a bounding sphere hierarchy to reduce the computation time of collision detection and it was successfully applied to the collision detection of a large numbers of objects. Teschner et al. [17] introduced an optimized spatial hashing based collision detection algorithm for dynamically deforming objects. Although many collision detection and response approaches for deformable objects have been studied, they still come up short on providing sufficient real-time performance and further improvement is imperative due to the ever increasing demand for high resolution 3D objects. In this paper, we extended our previous FFD-AABB algorithm [18] with an additional layer of the bounding sphere collision test. The proposed method quickly rejects and minimizes the full-blown FFD-AABB collision test to improve the overall speed of simulation under massive collision situations, and it is particularly advantageous for a densely meshed, large number of deformable objects.

2. Collision Handling with the FFD-AABB Algorithm

As the mesh complexity of deformable objects grows, the fast collision handling for those objects is still considered a significant challenge in real-time simulations. The FFD-based method is a one of the promising directions for alleviating those issues in 3D deformable objects animation and simulation. Barr [19] used a set of hierarchical transformations that include stretching, bending, twisting, and tapering operators to simply represent the broad sense of the deformation of objects. Sederberg and Parry [20] introduced the more general method that calculates the object deformation by deforming the outer lattice of embedded surface. Although the FFD-based method is restricted to uniform grids and its volume preservation is restricted, it generates smooth deformation and is quite easy to implement.

However, another impediment in achieving robust and realistic 3D deformation was the lack of efficient collision handling for the FFD-based method. Collision handling for deformable objects requires frequent information updating for the position of objects. Thus, collision detection and responses for the FFD-based surface of objects hinders the overall performance of the system. In addition, directly applying collision handling to FFD grids generates severe surface approximation errors and it causes critical and obvious visual floating artifacts between surfaces of objects [18]. Therefore, we previously proposed the FFD-AABB method, which is a dynamic bounding box, that deforms with the FFD grid along with the embedded surface meshes [18]. The FFD-AABB algorithm can be tightly integrated into an FFD-based complex deformable object simulation and the computational cost is cheaper than the traditional method because it requires only 8 node updates [18]. The spatial hashing approach is employed for fast culling by deforming the AABB.

FFD-AABB consists of a set of eight boundary nodes and they have local coordinates (*s, t, and u*) and global coordinates (*x, y, and z*) respectively. Fig. 1 shows a simple process of building FFD-AABB that consists of three steps. First, the FFD control grid is built for the target deformable object. Then AABB of the mesh is calculated for each FFD cell. Finally, the FFD local coordinates of boundary nodes and outside surface set are calculated for updating. Since the tri-linear interpolation is simple and fast, FFD-AABB can be updated quickly and independently from the complexity of the mesh. Since FFD frequently involves the scaling and shearing of objects, bounding spheres do not approximate the FFD surfaces of objects properly. Unlike FFD, FFD-AABB represents the embedded surfaces efficiently because FFD-AABB

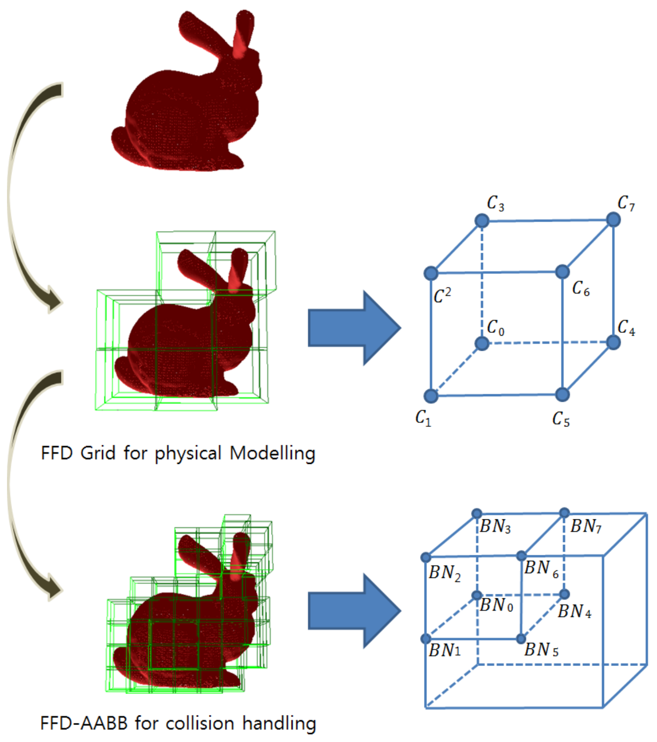


Fig. 1. The FFD-AABB building process

moves along with embedded meshes of deformable objects. The cost of an FFD-AABB update requires the evaluation of the eight node updates for each FFD cell. In addition, FFD-AABB tightly approximates the embedded surface of objects under the large deformation [10].

3. Activation of the FFD-AABB Based Collision Test using a Bounding Sphere

To improve the previous FFD-AABB based collision handling process, we analyzed the whole simulation process. Table 1 shows the computational cost of the experimental results for major FFD-AABB algorithm processes. Most of the computing time is spent in mesh updating and collision testing between objects. One of the big obstacles for FFD-AABB simulation is collision testing between objects and it takes around 57% of the computational time of the simulation, as shown in Table 1. Furthermore, the massive collision of densely meshed deformable objects causes even more computational costs for collision handling.

The FFD-AABB based collision test process requires performing the spatial hashing for determining PCP (potential collision pairs) in broad step and calculating the collision test between each node in the AABB of PCPs and each plane of the other AABB of PCPs in narrow step. To

Table 1. Results of the experimental test with the previous FFD-AABB algorithm (ms)

Object type	Number of objects	Mesh updating	Min, Max updating	Distance calculating	Collision test between the floor and objects	Collision test between objects	Total time
Alphabets	165	265.63	0.16	1.03	1.42	1061.39	1329.64
Spheres	100	115.19	0.10	0.47	1.25	216.96	334.28
Teapots	80	335.45	0.27	1.02	0.40	234.13	571.29
Buddhas	30	534.02	0.10	0.22	1.40	371.19	906.94

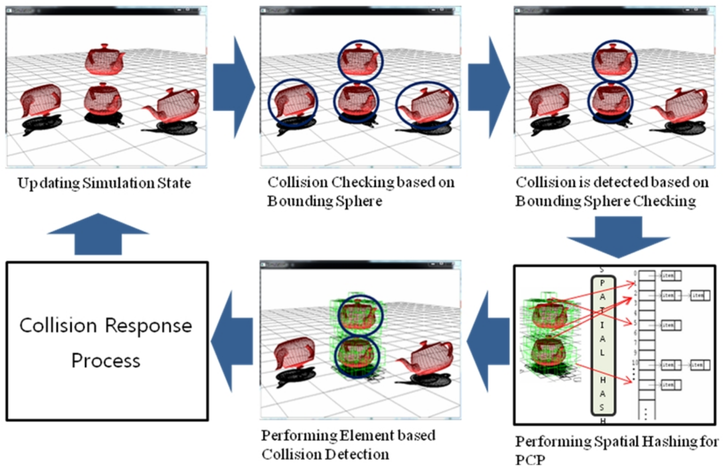

Updating Simulation State
Collision Checking based on Bounding Sphere
Collision is detected based on Bounding Sphere Checking
Collision Response Process
Performing Element based Collision Detection
SPATIAL HASH
Performing Spatial Hashing for PCP

Fig. 2. The proposed collision detection and response process

reduce the computational burden of the collision test, the proposed method introduced the bounding sphere based collision test before the broad and narrow collision test steps. Therefore, the proposed collision detection method consists of three phases. In the first phase, the bounding sphere based collision test is performed to quickly quit the unnecessary further collision detection using the simple distance calculation. In the second phase, spatial hashing is applied to determine the PCPs of the bounding box nodes. In the final phase, the bounding nodes in the same bucket of the hash table undergo detailed collision checking. Fig. 2 shows the proposed collision detection and response algorithm with additional bounding sphere based collision checking. Only collided objects that are detected by the bounding sphere collision test are progressed into the time-consuming element based collision detection step.

4. Experimental Results

The experimental test was performed on Intel Core i5 3.3GHz, 4GB RAM and the proposed algorithm was coded on Visual Studio 2010 under Window 7 OS. The proposed method was implemented in 3D using OpenGL. To measure the performance of the proposed method, we modeled four kinds of refined deformable objects such as letters of the alphabet, spheres, teapots, and Buddhas. These objects were freely falling down to the ground and they bumped against each other and the ground. Thus the simulations were carried out under complex collision environments, as shown in Fig. 3. The yellow circles are bounding spheres for quick rejection of further collision tests. Table 2 shows the number of vertices, triangles, and FFD cells for each object. Since letters of the alphabet have a different number of vertices, triangles, and FFD cells for each character, we used an average number of them for Table 2.

Fig. 4 shows the results of the experimental test based on FPS (Frame per Seconds) for each

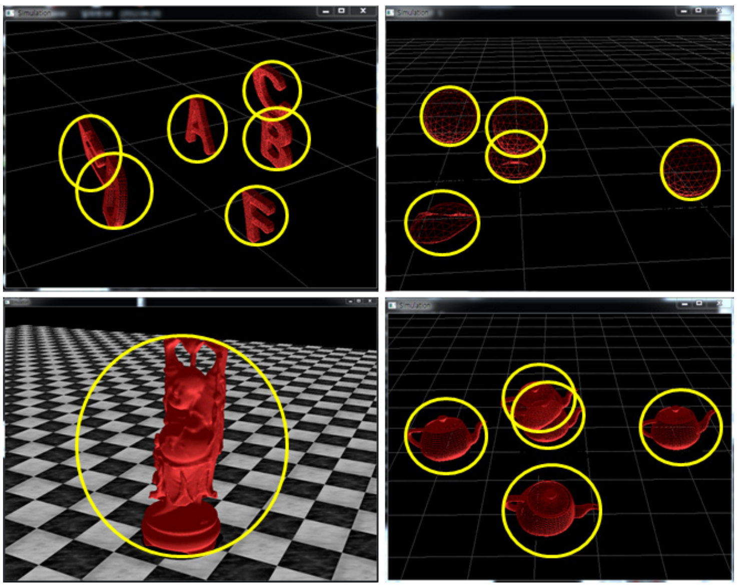
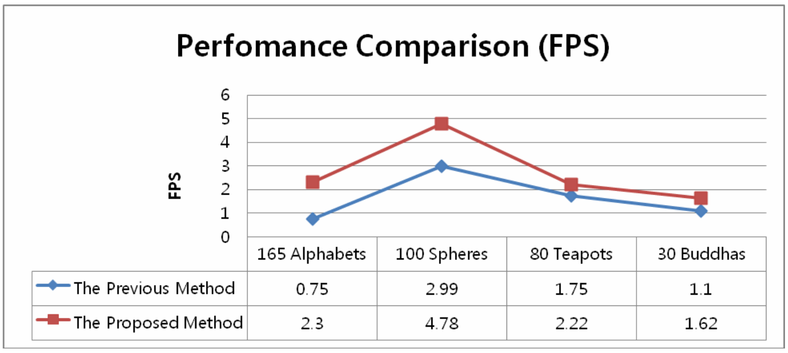


Fig. 3. The simulation examples using the propsoed method

experimental test. Although we applied a somewhat large number of detailed deformable objects to prove the robustness of the proposed algorithm for massive collision problems, the result returns around an 80% performance improvement rate compared to the previous method. Since the proposed algorithm quickly rejects the complicated collision test, it achieves better performance for more complicated examples that include more than 100 refined deformable objects. Fig. 5 depicts the simulation results for example models using the proposed algorithm.

Table 2. The number of components for each refined deformable object

Object type	Number of Vertices	Number of Triangles	Number of FFD cells
Letters of the alphabet	1,141	2,279	34
Spheres	162	320	38
Teapots	3,242	6,400	84
Buddhas	32,328	67,240	148



	165 Alphabets	100 Spheres	80 Teapots	30 Buddhas
The Previous Method	0.75	2.99	1.75	1.1
The Proposed Method	2.3	4.78	2.22	1.62

Fig. 4. Result of the performance comparison based on FPS

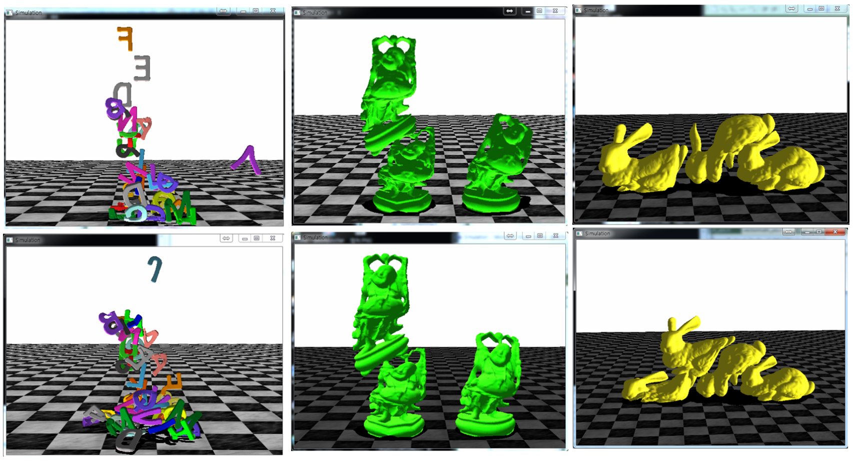


Fig. 5. Snapshots of simulation results

5. Conclusion

In this paper, we extended the previous FFD-AABB algorithm to provide an enhanced performance of massive collision handling between densely meshed deformable objects. We added a bounding sphere collision test to the previous FFD-AABB algorithm to promptly remove the unnecessary time-consuming collision test. The computation cost for the proposed method was 80% faster than the previous FFD-AABB algorithm. The mesh updating process, which is another computational burden for the FFD-AABB algorithm, can be modified and updated to apply the real-time applications in future work.

References

[1] S. F. Gibson and M. Brian, *"A survey of deformable models in computer graphics. Technical Report TR-97-19," Mitsubishi Electric Research Laboratories*, Cambridge, MA, November, 1997.

[2] D. Breen, D. House, and P. Getto, *"A physically-based particle model of woven cloth," The Visual Computer*, Vol.8, 1992, pp.264-277.

[3] J. Collier, B. Collier, G. O'Toole, and S. Sargand, *"Drape prediction by means of Finite-element analysis," Journal of the Textile Institute,* 1991, pp.96-107.

[4] K. Waters, *"A physical model of facial tissue and muscle articulation derived from computer tomography," In Proceedings of Visualization in Biomedical Computing*, Vol.1808, 1992, pp.574-583.

[5] M. Bro-Nielsen, *"Surgery Simulation Using Fast Finite Elements," Proc. Visualization in Biomedical Computing*, Springer-Verlag, 1996, pp.529-534.

[6] J. Chadwick, D. Haumann, and R. Parent, *"Layered construction for deformable animated character," In Computer Graphics Proceedings, Annual Conference Series, Proceedings of SIGGRAPH 89, ACM SIGGRAPH*, 1989, pp.243-252.

[7] D. L. James and D. K. Pai, *"ArtDefo: accurate real time deformable objects", Proceedings of SIGGRAPH 99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999, pp.65-72.

[8] P. Faloutsos, M. Panne and D. van de, *"Dynamic free-form deformations for animation synthesis," IEEE Trans. Vis. Comput. Graph.,* Vol.3. 1997, pp.201-214.

[9] S. Capell, S. Green, B. Curless, T. Duchamp and Z. Popovic, *"Interactive skeleton-driven dynamic deformations," Proc. ACM SIGGRAPH 02,* 2002, pp.586-593.

[10] A.R. Rivers and D.L. James, *"FastLSM: fast lattice shape matching for robust real-time deformation," Proc. ACM SIGGRAPH 07*, 2007, pp.82.

[11] M. Teschner, S. Kimmerle, G. Zachmann, B. Heidelberger, L. Raghupathi, A. Fuhrmann, M.P. Cani, F. Faure, N. Magnetat-Thalmann and W. Strasser, *"Collision detection for deformable objects," In: Eurographics State-of-the-Art Report (EG-STAR)*, 2004, pp.119-139.

[12] C. Ericson, *"Real-time Collision Detection," Elsevier*, 2005

[13] P. Jimenez, F. Thomas and C. Torras, *"3D collision detection: a survey,"* Comput. Graph., Vol.25, 2001, pp.269-285.

[14] P.M. Hubbard, *"Collision detection for interactive graphics applications," IEEE Trans. Vis. Comput. Graphics,* Vol.1, 1995, pp.218-230.

[15] G. van den Bergen, *"Efficient collision detection of complex deformable models using AABB trees," Graphics Tools,* Vol.2, 1997, pp.1-13.

[16] D.L. James and D.K. Pai, *"BD-tree: output-sensitive collision detection for reduced deformable models," ACM Trans. Graph. (SIGGRAPH'04),* Vol.23, No.3, 2004.

[17] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets and M. Gross, *"Optimized spatial hashing for collision detection of deformable objects," Proc. Eighth Int. Fall Workshop Vision, Modeling, and Visualization (VMV'03),* 2003, pp.47-54.

[18] S. Jung, M. Hong and M. Choi, *"Collision Handling for Free-Form Deformation Embedded Sur-*

face," IET Image Processing, Vol.5, 2011, pp.341-348.

[19] A. H. Barr, *"Global and Local Deformations of Solid Primitives," Proceedings of SIGGRAPH 84, Computer Graphics 18*, No.3, July, 1984, pp.21-30.

[20] T. W. Sederberg and S. R. Parry, *"Free-Form Deformation of Solid Geometric Models," Proceedings of SIGGRAPH 86, Computer Graphics 20*, No.4, August, 1986, pp.151-159.

Jae-Hong Jeon

He received the BS degrees in Computer Software Engineering from Soonchunhyaung University in 2012. Now he is undertaking a master degree of computer engineering courses as a member of the computer graphics lab at Soonchunhyang University. His research interests are game development, computer graphics, AR (Augmented Reality) and embedded motion capture.

Min-Hyung Choi

Prof. Choi received his M.S. and Ph.D. from University of Iowa in 1996 and 1999 respectively. He joined the Computer Science and Engineering Department at the University of Colorado at Denver in 1999. Currently he is the Director of Computer Graphics and Virtual Environments Laboratory. His research interests are in Computer Graphics, Scientific Visualization and Human Computer Interaction with an emphasis on physically-based modeling and simulation for medical and bioinformatics applications.

Min Hong

He is an assistant professor in the Department of Computer Software Engineering at Soonchunhyang University, South Korea. He received an MS in computer science from the University of Colorado at Boulder and a PhD in bioinformatics from the University of Colorado at Denver and Health Sciences Center. His research interests include Computer graphics, Physically-based modeling and simulation, Image Processing, and Computer game.