Free-Form Deformation Axis Aligned Bounding Box

Sunhwa Jung[1], Min Hong[2], and Min-Hyung Choi[1]

[1] Department of Computer Science and Engineering, University of Colorado Denver, Campus Box 109, PO Box 173364, Denver, CO 80217, USA
sunhwa.jung@email.ucdenver.edu, min.choi@ucdenver.edu
[2] Division of Computer Science and Engineering, Soonchunhyang University, 646 Eupnae-ri Shinchang-myeon Asan-si, Chungcheongnam-do, 336-745, Korea
mhong@sch.ac.kr

Abstract. We present a new efficient collision-handling technique of free-form deformation (FFD) of an embedded surface. By adapting FFD, modeling deformation has been substantially simplified to make possible interactive rate animation of a deformable object even for a complex embedded mesh. However, the lack of effective collision detection and resolution schemes for an FFD-embedded surface hinders the overall performance and often becomes a bottleneck. Most existing collision handling techniques can be directly applied to an FFD grid for fast computation, but surface approximation error would be apparent and it could cause noticeable visual artifacts. On the other hand, applying collision detection and resolution techniques directly to the embedded surface is extremely expensive and can obliterate the viability of real-time FFD simulation because the embedded surface has a high resolution in most cases. We present a fast collision detection and resolution method for the embedded surface in an FFD-enhanced simulation maintaining the approximation error of the embedded surface. Our techniques for detection and resolution provide an ability to balance speed and quality.

Keywords: Free-Form Deformation, Collision Detection and Resolution.

1 Introduction

Recent advances in real-time deformable object simulation have led to interactive animation of high resolution meshes becoming a viable solution for many applications. A fast deformation computation was done using reduced deformable models [4], but the degree of freedom for a deformable object is limited, and modeling for such a structure is still not trivial. A promising direction for a fast deformable object animation is based on a free-form deformation (FFD) grid placed over a complex embedded surface [7], providing efficiency and flexibility without placing any limitation on the underlying geometric mesh. However, one of the most prominent downsides of the FFD grid approach is that it lacks a proper collision and contact resolution scheme pertinent to the deformation solution method. Applying a conventional collision handling method to the FFD grids generates severe surface approximation error resulting in significant visual artifacts of floating. On the other hand, performing a primitive level of exact collision detection for complex objects is not practical and can eliminate the benefits of having FFD-based simulation in terms of computational cost.

We present efficient collision detection and resolution method based on a free-form deformation axis aligned bounding box (FFD AABB) to support fast collision detection and resolution for the FFD grid-based simulation.

Our specific contributions are:

- new FFD AABB techniques for an efficient collision detection and resolution that can be tightly integrated into an FFD embedded surface simulation, and
- fast culling in broad-phase collision detection using spatial hashing of deforming AABB that does not require any modification of structure in topological changes such as fracturing and cutting.

2 Related Work

Free-form deformation-based simulation is well known for its ease of modeling and fast real-time performance. Free-form deformation has been already employed for real-time deformation of a complex geometry based on the mass-spring system, finite element method [1], [2], [3], [8] and lattice shape matching [6]. Although it limits the degree of deformation freedom, uncomplicated adaptation to physically-based simulations makes the overall modeling process simple. Especially for the applications that do not require accuracy (e.g., animation and gaming), it is a sensible choice over sophisticated yet slower methods.

Collision detection and resolution of deformable objects has been intensively studied and significant progress has been made in accuracy and speed [10], [11], [12]. But no collision handling technique for the FFD-embedded surface has been proposed yet to our knowledge. Lately fast collision detection methods using bounding sphere tree employing deformation bounded update techniques are proposed. Bounded Deformation Tree (BD-tree) [5] updates bounding spheres according to the functions of the reduced coordinates independently from geometry for reduced deformable objects. An efficient bounding sphere hierarchy update technique [9] is proposed for matrix-based deformation. These approaches are appropriate for the aimed deformation techniques, but it is not trivial to use them for FFD-based simulation because FFD-based deformation is not represented by a matrix or reduced coordinates. In addition bounding sphere can easily overestimate an FFD-embedded surface without re-estimating the radius for all the primitives. In contrast to the previous methods, our method tightly estimates the embedded surface and similarly handles massive collision detection efficiently without any expensive hierarchy update costs.

3 Free-Form Deformation Axis Aligned Bounding Box

3.1 FFD AABB for Embedded Objects

An Axis aligned bounding box (AABB) consists of six boundary float variables (maximum x, minimum x, maximum y, minimum y, maximum z, and minimum z) that are updated at each time step by traversing the primitives in the AABB. In general the cost of an AABB update cost is at least O (N). Unlike AABB, FFD AABB consists of a set of eight boundary nodes (BN) with local coordinates (s, t, and u) and a set of outside surfaces (OS) that represent the surface of AABB (see Eq. 1).

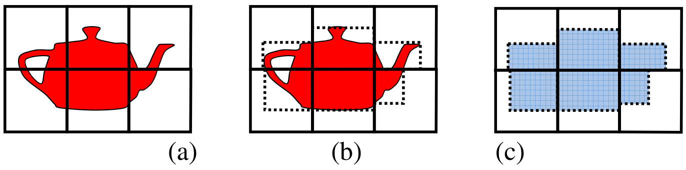

Fig. 1 FFD AABB building process: voxelizing (a), building AABB of voxels (b), and calculating the FFD local coordinates of boundary nodes for BN and set OS (c)

Fig. 1 illustrates FFD AABB building process. Each BN is updated at each time through FFD. Since six surfaces exist for one FFD cell and the orientation of AABB is fixed (see Fig. 2), six Boolean variables can represent an OS set. Inner FFD AABBs are equivalent to their grid cells and can be excluded from the collision handling process. A BN is updated at each time step using an FFD formulation along with embedded surface nodes. As a result, FFD AABB contains the embedded surfaces of an FFD cell while the cell undergoes dynamic deformation.

$$FFD\ AABB = \{BN,\ OS\} \tag{1}$$

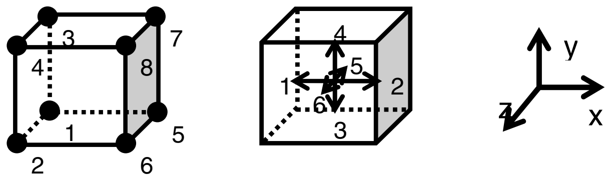

Fig. 2. FFD AABB Components: Eight BNs and six OSs are indexed matching to FFD cell nodes (left) and FFD cell surfaces (middle), respectively. The index is preset according to the axis x, y, and z (right).

3.2 FFD AABB for Deformed Embedded Objects

According to the movements of control points in the FFD grid, the embedded objects may go through various deformations. Since FFD involves scaling and shearing frequently, bounding spheres do not approximate the FFD embedded surfaces, but FFD AABB represents the embedded surfaces efficiently. As compared to the conventional FFD algorithm, the cost of an FFD AABB update is only adding eight-nodes update to the embedded surface nodes updates. Figure 3 shows a relationship between the embedded surfaces, FFD AABB, and the FFD grid.

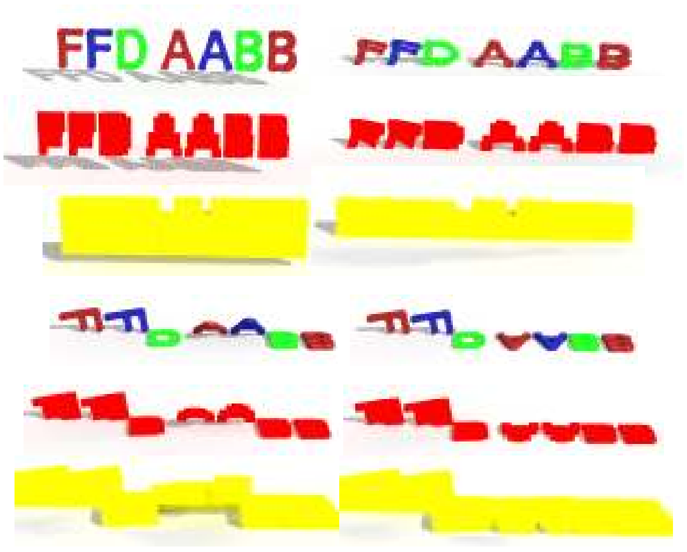

Fig. 3. The embedded surface of letters (first and forth lines), FFD AABB (second and fifth lines), and FFD Grid (third and sixth lines), under free-form deformation: FFD AABB is used for collision detection and resolution and the FFD grid is used to simulate objects. FFD AABB tightly holds the embedded surface.

4 Collision Detection and Resolution

Our method consists of two phases. In the first phase spatial hashing is applied to determine the potential collision pairs of FFD AABB. In the second phase FFD AABBs in the same voxels undergo collision checking and collision correction at the same time. Typically collision detection is separate from collision response. Collision detection returns the collision result, such as a list of the collided primitive pairs (e.g., bounding sphere/bounding sphere). Then the collision response is calculated and applied to the pair. In this process, one of the most time-consuming steps is to remove duplicated collision result in order to avoid applying multiple collision corrections. The key idea of our method is to apply the collision correction and update the FFD AABBs of the corrected cells at the collision detection time. Instead of returning a list of potential collision pairs (PCP) after performing collision detection, the collision response is applied and the state of the FFD AABB is updated. Removing redundant pairs from the list of PCPs is expensive when the number of collisions is relatively large. To remove the repetition of collision detection and resolution for the same pair, collision-involved FFD AABBs are updated after the collision correction has been applied. This guarantees that the state of FFD AABB is synchronized throughout the whole collision detection process.

4.1 Spatial Hashing of Deforming AABB

In the broad phase of collision detection, bounding volume hierarchy or spatial subdivision can be used, and we chose spatial subdivision similar to the optimized spatial hashing [11] that supports topological changes efficiently. Axis aligned bounding box of FFD AABB (deforming AABB) is used for spatial hashing. Deforming AABB is easily calculated at each time step for FFD AABB and spatial hashing performs on deforming AABBs. After performing voxelization of deforming AABB, PCP is determined. For optimal performance of spatial hashing, the size of the spatial hashing cell should be close to the largest length of the edges in optimal hashing. For our case, the FFD cell size is proper for the size of spatial hashing cell because the FFD cell is the spatial hashing primitive. We used 4,000 for the hash table size because it gives the optimal performance. Spatial hashing time is the constant (4 ms) throughout the simulation when the number of objects is the same even if the number of collisions is increasing.

4.2 FFD AABB Collision Detection

In narrow phase collision detection, FFD AABB of PCP is checked for collision. We performed point to FFD AABB and surface to FFD AABB collision checks. A and B are the FFD AABB in the same PCP. If one of BNs of A is inside of all OSs of B or vice versa, the possibility of a collision is determined by evaluating the plane equations for the OSs. In the case of a collision, the result is BNs inside of the other FFD AABB. For a surface to FFD AABB collision check, the center of mass point is calculated and performed point/FFD AABB collision check. The collision detection can be made more accurate by adding more points from the surface into the point to FFD AABB collision check. This scheme can be used as another level of culling for exact collision detection between embedded surfaces, but we applied the response based on the collision detection result between point and FFD AABB. If the number of the FFD AABB for a simulated object is large, the approximation of FFD AABB is accurate.

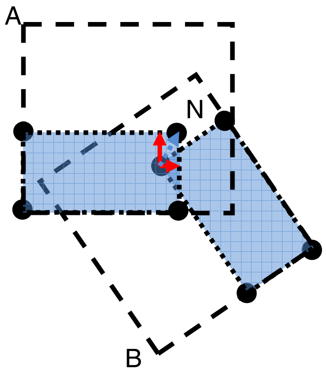

Fig. 4. Calculating collision normal: The collision normal is the average surface normal vectors of OS

4.3 Impulse Response

For collision response, we use impulse-based collision correction and geometric corrections. Collision normal (N) is the average of the surface normal vectors of OS weighted with the penetration depth (Eq. 2). The velocity of the cell node i ($V_{i\in A}, V_{i\in B}$) of A and B is corrected according to N . The response velocity ($\vec{V}_{i\in A}, \vec{V}_{i\in B}$) is calculated using Eq. 3 and applied to the cell nodes in case the relative velocity between A and B is negative (getting closer).

$$N = \frac{\sum_{S\in OS} D_S N_S}{\left\| \sum_{S\in OS} D_S N_S \right\|}, \tag{2}$$

where D_S is the penetration depth and N_S is the surface normal of S .

$$\begin{aligned} \vec{V}_{i\in A} &= \alpha(V_{i\in A}\cdot(-N))(-N) + \beta(V_{i\in A} - (V_{i\in A}\cdot(-N))(-N)) \\ \vec{V}_{i\in B} &= \alpha(V_{i\in B}\cdot N)N + \beta(V_{i\in B} - (V_{i\in B}\cdot N)N) \end{aligned} \tag{3}$$

4.4 Geometric Correction for Embedded Surface

Geometric correction (C_A and C_B) is half of the minimum penetration depth (Eq. 4) among OS in the opposite direction (see Fig. 5). To correct an embedded surface, the correction is applied to the nodes of the cells (A and B) matching to the penetrated FFD AABB. After applying geometric correction, FFD AABB must be updated, and the next collision correction will be calculated correctly.

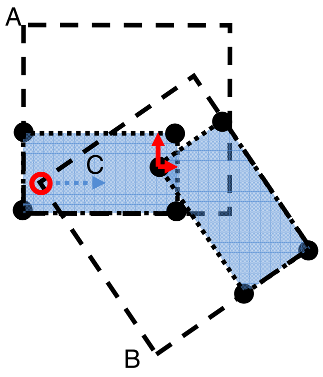

Fig. 5. Geometric correction: the surface normal vectors of OS are compared to get minimum depth penetration

$$C_A = \frac{1}{2}\min(N_S, S \in OS_A)$$
$$C_B = -\frac{1}{2}\min(N_S, S \in OS_A) \tag{4}$$

5 Results

We checked the accuracy of surface proximity of FFD AABB and the performance of overall collision detection through simulation of many different shapes of objects using fast lattice shape matching. The example simulations were run on a PC with Pentium D 3.0 and 2.99 GHz CPU and 2G ram.

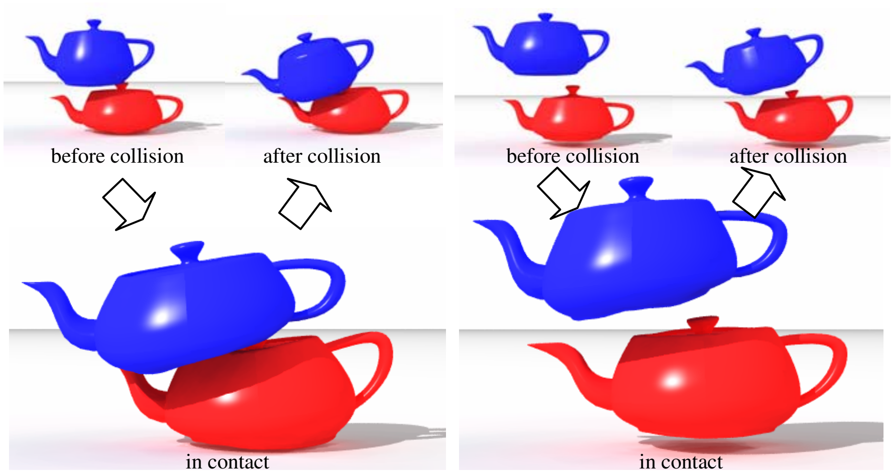

Fig. 6. Stacking two teapots: snapshots of collision handling with FFD AABB (left), and the snapshots of collision handling with an FFD grid (right). The surface estimation of FFD AABB is more accurate than the FFD grid. The teapots are staked without visual artifacts such as penetration or a gap between objects.

Teapot simulation: To test if our method produces close proximity of the embedded surface between complex curved and non-convex surfaces, we simulated teapots. Each teapot in Fig. 6 has 3,644 nodes, 6,320 triangles, and 602 FFD grid cells used for simulation. FFD AABB deforms with FFD grid and contains the embedded surface mesh more tightly than FFD grid. The proximity of FFD grid for the embedded surface is too rough. Results of collision detection by FFD grid show a large gap between the embedded surface meshes and between the embedded surface mesh and the floor. On the other hand the proximity between contact surfaces are tightly maintained by our method and doesn't show floating artifacts. Still both the collision handling by the FFD grid and the collision handling by FFD AABB manages 1.7 ms for collision detection and response altogether.

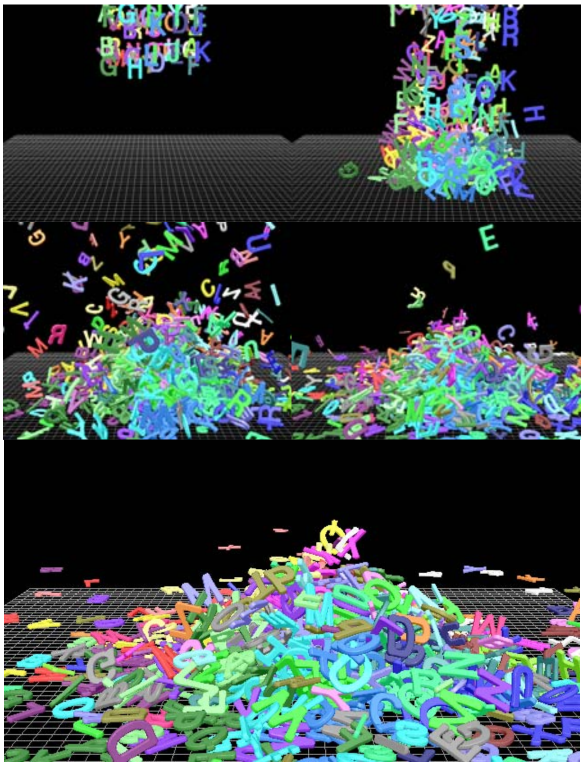

Fig. 7. Alphabet letters simulation: the snapshots of 1014 alphabet letters simulation at 165 (top left), 825 (top right), 1605 (middle left), 2280 (middle right), and 3225 (bottom)

Alphabet simulation: To verify whether our method can handle massive collisions, where exact contact is not called, we simulated 1,014 alphabet letters. The deforming shapes of 26 alphabets are various and FFD AABB efficiently holds the embedded objects. Each alphabet model is dropped with a group toward a confined area to create a large number of sustained collisions and contacts. Initial conditions for velocity and orientation or torque for the objects are randomly generated. Figure 7 shows a series of snapshots of the simulation time-stamped as it progresses. The typical FFD grid for each alphabet model is between 8 to 16 depending on the shape and topology of the model. The number of objects in Fig. 7 is 1,014 and the number of triangles is roughly three million. Initially the number of collision is relatively small but as alphabets piled up the total number of collision/contact grows accordingly. Processing time per collision/response is kept relatively constant. Our method scales up reasonably as the collision handling instances are increased. Our procedure manages the collision handling process within an interactive rate at 30 FPS up to 300 collisions (see Fig. 8). The average number of nodes in the embedded surface mesh of alphabets is 1,118 ranging from 766 to 1,434. The average number of triangles is 2,234 ranging from 1,528 to 2,872. The average collision handling time is 279 MS throughout the whole simulation.

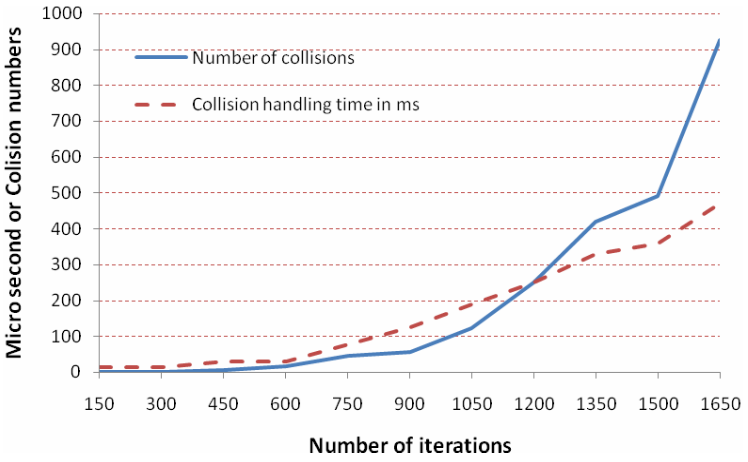

Fig. 8. Collision handling performance and the number of collisions for the alphabet simulation

6 Conclusion

Free-form-deformation-based simulation can be utilized in many areas, but collision detection and resolution can be a bottleneck of the simulation. We propose FFD AABB which deforms along with the embedded surface and its collision detection and resolution method, which performs fast and efficiently with FFD-based simulation techniques. Our technique has a few limitations. Our method is meant for FFD-based simulation and only node/FFD AABB and the center of mass/FFD AABB collision checks are performed, so edges can penetrate each other. But it does not produce a serious artifact in our example simulations. Adaptively adding more nodes in the edges for the collision check will treat the slight artifacts without changing the algorithm with a little bit of additional computational cost. Self-collision, excluded in this paper, will be the future direction of this project.

References

1. Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z.: Interactive Skeleton-Driven Dynamic Deformations. In: Proc. ACM SIGGRAPH 2002, pp. 586–593. ACM, New York (2002)
2. Capell, S., Green, S., Curless, B., Duchamp, T., Popović, Z.: A Multiresolution Framework for Dynamic Deformations. In: Proc. the 2002 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation (SCA 2002), pp. 41–47. ACM, New York (2002)
3. Frisch, N., Ertl, T.: Deformation of Finite Element Meshes Using Directly Manipulated Free-Form Deformation. In: Proc. the Seventh ACM Symposium on Solid Modeling and Applications (SMA 2002), pp. 249–256. ACM, New York (2002)
4. James, D.L., Pai, D.K.: Dyrt: Dynamic Response Textures for Real Time Deformation Simulation with Graphics Hard-ware. In: Proc. ACM SIGGRAPH 2002, pp. 582–585. ACM, New York (2002)

5. James, D.L., Pai, D.K.: BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models. ACM Trans. Graph (SIGGRAPH 2004) 23(3) (2004)
6. Rivers, A.R., James, D.L.: FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation. In: Proc. ACM SIGGRAPH 2007, p. 82. ACM, New York (2007)
7. Spillmann, J., Becker, M., Teschner, M.: Efficient Updates of Bounding Sphere Hierarchies for Geometrically Deformable Models. J. Vis. Comun. Image Represent. 18(2), 101–108 (2007)
8. Sederberg, T.W., Parry, S.R.: Free-Form Deformation of Solid Geometric Models. SIGGRAPH Computer Graphics 20(4), 151–160 (1986)
9. Sela, G., Subag, J., Lindblad, A., Albocher, D., Schein, S., Elber, G.: Real-Time Haptic Incision Simulation Using FEM-Based Discontinuous Free Form Deformation. In: Proc. the 2006 ACM Symposium on Solid and Physical Modeling (SPM 2006), pp. 75–84. ACM, New York (2006)
10. Sud, A., Govindaraju, N., Gayle, R., Kabul, I., Manocha, D.: Fast Proximity Computation among Deformable Models Using Discrete Voronoi Diagrams. ACM Trans. Graph. 25(3), 1144–1153 (2006)
11. Teschner, M., Heidelberger, B., Müller, M., Pomeranets, D., Gross, M.: Optimized Spatial Hashing for Collision Detection of Deformable Objects. In: Proc. the 8th Int'l Fall Workshop Vision, Modeling, and Visualization (VMV 2003), pp. 47–54 (2003)
12. Teschner, M., Kimmerle, S., Zachmann, G., Heidelberger, B., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnetat-Thalmann, N., Strasser, W.: Collision Detection for Deformable Objects. Computer Graphics Forum 24(1), 61–81 (2005)