A Novel Method for Cloth-Body Collision Detection

Xiaolong Zhu[1,2], Shihong Xia[1], Yong Yu[1,2], and Tianlu Mao[1,2]

[1] Institute of Computing Technology, Chinese Academy of Sciences, 100080 Beijing, China
[2] Graduate School of the Chinese Academy of Sciences, 100080 Beijing, China
{zhuxiaolong,xsh,yyu,ltm}@ict.ac.cn

Abstract. This paper presents a novel cloth-body collision detection method by using the generalized cross-sectional contour technique, which has two main steps. During preprocessing step, the so-called skin hierarchical structure (Skin-H) of the body is constructed by using the improved generalized cross-sectional contour technique, which doesn't need to be updated in subsequent step. During runtime step, the cloth vertices are projected onto Skin-H structure efficiently, and then the exact collision detection can be done by a ray-triangle test technique at the lowest level of the structure. The simulation result demonstrates that the proposed method has some advantages in algorithm's efficiency, accuracy as well as practicability.

Keywords: Collision Detection, Hierarchical Approaches, Image-Based Methods, Cloth Simulation, Animation.

1 Introduction

Cloth simulation has been an area of active research in the past 15 years due to its increasing importance for commercial applications such as virtual try-on [2,3,9,20]. Unfortunately, cloth-body collision detection is the crucial but expensive technology and proved to be the bottleneck of dynamic cloth simulation. Motivated by this fact, numerous approaches have been proposed to achieve efficiency and accuracy as well as practicability. These approaches can be classified into: (1)*object-space interference test*[2,3,4,11,12,19], and (2)*image-space interference test*[1,5,6,7,8,14,20]. Most of object-space methods use bounding volume hierarchy (BVH) to accelerate interference computations. For example, An OBB-Hierarchy is used in [19] to represent polyhedra with triangulated boundaries. The overlaps between OBBs are determined by performing 15 simple axis projection tests, however, it associates with high cost of building and updating hierarchy. [11] improved the efficiency of BVH by proposing adapted techniques for building and traversing hierarchies, and [4] introduced the Bounded Deformation Tree to perform collision detection for reduced deformable models at similar costs to standard algorithms for rigid objects. In addition, with the recent advent of high performance GPU, the GPU-based methods have opened new avenues for improving the efficiency of collision detection. The methods are based on projecting the object geometry onto the image plane and performing the analysis in a dimensionally reduced space with the depth map maintained in an image buffer such as the Z-buffer. The first application of these methods to dynamic cloth simulation has been presented in [20]. In this approach, an avatar is rendered from a front and a back view to generate the depth information for collision query. Recently, more and more hybrid methods which combined object-space methods with GPU-based ones were presented, such as [1, 8, 13]. In [8], it introduced $(\pi,\beta)-\text{surface}$ to dynamically generate a hierarchy of cloth bounding boxes in order to perform object-level culling and GPU-based intersection tests using conventional graphics hardware support. Another kind of GPU-based methods is recoding the geometry information into textures and then computing the collision results in the fragment program [1, 5]. There are many other notable approaches [15, 16] proposed to improve the efficiency and accuracy of collision detections. For example, a chromatic mesh decomposition is used to develop a linear time detection algorithm in [15].

By comparison, GPU-based collision detection methods are easy to implement and most of which involve no preprocessing and therefore can be applied to both rigid and deformable models. Meanwhile, it has a good potential to exploit the hardware-assisted rendering features of the current graphics boards. The main problem of GPU-based methods, however, is that it suffers from limited precision which is due to the limited viewport resolution and GPU float-point errors [6, 14]. Most object-space methods work well for rigid objects; nevertheless, they need frequently update the hierarchy especially for deformable models. In this paper, a novel hierarchy named skin hierarchical structure (Skin-H) is presented, which is different from the conventional BVH . It can avoid time consuming hierarchy updating because the new hierarchical building method considers the special structure of human and cloth.

2 Collision Detection Using Skin-H

In this section, we give an overview of our collision detection algorithm. It includes the building of Skin-H during precomputing and collision detection in runtime. Without loss of generality, we assume that the body model is seamless and can be represented as triangular meshes, which are driven by the corresponding skeleton. We firstly proposed a so-called skin hierarchical structure (Skin-H), according to the driving principle of virtual human with seamless model and the fact, that the displacement of skin is very small related to the innate bone. The Skin-H is invariant while virtual human moving. It's needless to update the hierarchy frequently. This is because each human skin vertex is tied to a certain joint to generate joint block which can be treat as a rigid object. Although there are skin deformations, they can also be handled well in this paper. Subsequently each cloth vertex is projected onto Skin-H, and ray-triangle collision test can be used during runtime.

2.1 Skin Hierarchical Structure (Skin-H)

The design choice of the type of bounding volume (BV) mainly determines the efficiency of traversing and updating the BVH. People have tried many types of BV such as OBBs and DOPs attempted to quickly update or refit the BVH after a deformation has taken place. However, the hierarchy should be updated after each frame for animation cloth.

To solve this problem, we design our hierarchy based on the principal of skeleton driven deformation and the structure of the special human body shape. The skeleton deformation uses vertex weights which tie the joints and the mesh together. This could help dividing the human skin triangles into several **Blocks** according to different joints influence different skin vertices. For each **Block**, the local coordinate system is built and then each block is partitioned further into **Segments** and **Sections**.

Block. Block is the top of Skin-H. Two methods (Fig.1) for segmenting the human skin into blocks have been tried in our experiment. One is to select several main joints and take them as the boundaries of the blocks which are built according to the coordinate of each skin vertex. Another method for classifying the vertices is based on the seamless human model, which is driven by the articulated skeleton, wherein each skin vertex is affected by the corresponding joints. So we can classify all of the skin vertices by their effective weights.

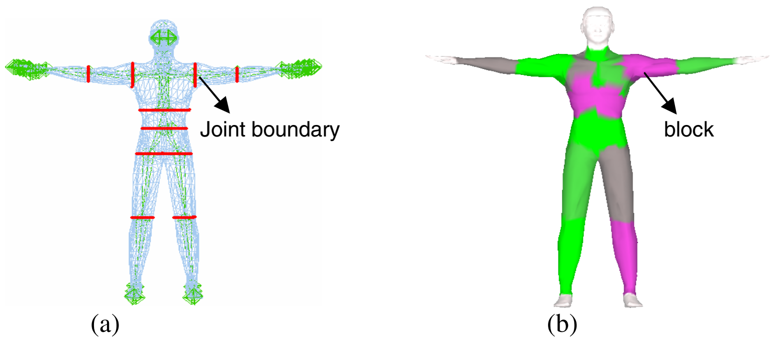


Fig. 1. Two methods for generating Blocks: (a) take the joints as the boundaries of blocks;(b) classify vertices based on effective weights

The former method mainly have two shortcomings: 1) it's difficult to project the cloth vertex into the corresponding block when the human is in motion because there are some boundary planes would overlap each other then; 2)it doesn't consider the skin deformation which could result in missing collisions near the boundary. This paper therefore adopts the latter method which classifies the skin vertex based on the influence weights. Generally, there are tens of joints in a human body skeleton. It is needless to divide the human skin into so many blocks. The 13 main joints are selected for efficiency consideration. For each main joint, because its translation and rotation transformation will affect a number of skin vertices, a weight threshold δ_{weight} is set in order to determine each vertex belonging to which block. As a result, the joint block generation algorithm is described as follows.

Algorithm 1. Generating Block

```
For each main joint joint_i
  For each skin vertex vertex_j affected by joint_i
      If weight(joint_i, vertex_j ) >= $\delta_{\text{weight}}$
            AppendToBlock( joint_i, GeTriangle (vertex_j ) );
```

In the algorithm 1, GeTriangle (vertex_j) returns the triangle which contains the vertex vertex_j and the threshold δ_{weight} is determined keeping to the following two rules:

1) Eliminate the influenced vertices far away from the joint;
2) Generate redundancy in the adjacent blocks against missing collisions during detection.

Segment. The Skin-H is built by using top-down strategy. The segment is on the second level of the Skin-H. There are usually hundreds of triangles in a joint block. For reducing the times of the ray-triangle collision detection, each joint block is divided further into segments.

The method of generating segments is based on the generalized cross-sectional contour technique [10,18]. For each block, we choose the joint center as the origin and set the direction of y-axis along the corresponding bone to build local coordinate system (Fig. 2). As a result, we can easily get the local coordinate and the polar angle of the center of each triangle which belongs to this joint block. The local y-coordinate helps us divide the block into several segments.

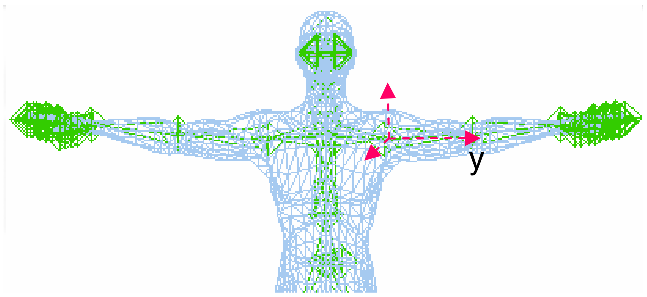


Fig. 2. Building local coodinate system for each segment

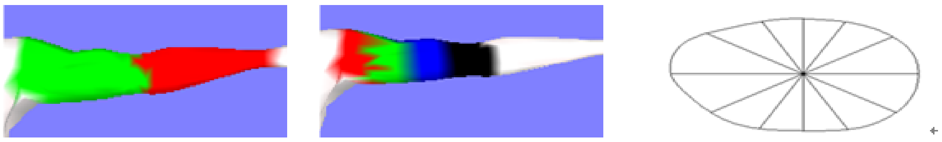
Section. Sections are in the bottom of the Skin-H and built according to polar angle. Every segment could be considered as a column so that the triangles in them can be classified into several sections in term of the polar angles.

In pretreatment, all of the skin triangles are classified into sections. Each level of the Skin-H stores the indices of the corresponding skin triangles. For avoiding missing the collisions result from skin deformation, a certain degree of redundancy is kept within each level. These measures guarantee it's needless to refit or update the Skin-H during runtime.

As shown in Fig.3, the Skin-H has three levels (see figure 1): *Block*, *Segment* and *Section*.

2.2 Project Cloth Vertices on Skin-H

In runtime, our cloth-body collision detection algorithm includes coarsely localizing the joint blocks and the segments and further the sections which potentially collide with cloth vertices and at last performing the ray-triangle intersection test.



(a) Block (b) Segment (c) Section

Fig. 3. The three levels of the Skin-H

There is possibility that each cloth vertex collides or even penetrates the skin triangles. So we traverse each cloth vertex and search skin triangles which potentially collide with the vertex. These triangles must be in a certain section, and located via projecting the vertex to the Skin-H.

The first step of searching the potential colliding skin triangles is that locate the potential joint blocks (PCB). In our experiment, we choose 13 main joints and classify the all skin triangles into 13 joint blocks in pretreatment. Meanwhile, the maximal distance $dis_{\text{bone\_i}}$ from the triangles to the corresponding bone is calculated for each joint block. During collision detection, the distance $dis(\text{vertex\_i}, \text{bone\_i})$ from cloth vertex to each main bone is computed and the joint block is added to the PCB if $dis(\text{vertex\_i}, \text{bone\_i})$ is less than $dis_{\text{bone\_i}}$. Locate the potentially colliding segments and sections in Skin-H just need to transform the global coordinate of the cloth vertex to the local coordinate under the joint block's coordinate system. The local y-coordinate and the local polar angle can help to find the potential Segments and Sections. The whole projection or location method is described as follows.

Algorithm 2. Cloth-vertex Projecting

```
For each cloth vertex VCloth
• Computing the distance from cloth vertex to each block
• Traverse the blocks to find blocks satisfying $dis(\text{vertex\_i}, \text{bone\_i}) < dis_{\text{bone\_i}}$ and add them
  to PCB
    For each block $\in$ PCB
        • Vcloth_localCoord = GetLocalCoord( VCloth, Block_i )
        • segment = GetSegment( Vcloth_localCoord, Block_i )
        • section = GetSection( Vcloth_localCoord,segment )
            For each triangle Tri_i in section
                • RayTriangleCollisionTest( VCloth, Tri_i )
```

In this algorithm, *GetLocalCoord(VCloth, Block_i)* calculate the local coordinate includes polar angle of the cloth vertex, and then, *GetSegment(Vcloth_localCoord, Block_i)* and *GetSection(Vcloth_localCoord, segment)* locate the segment and the section in Skin-H according to the local y-coordinate and polar angle.

2.3 Ray-Triangle Intersection Detection

For each triangle in the section above, Moller's method [21] is used for Ray-Triangle intersection test and computing the penetration depth. The difficulty is that obtain a

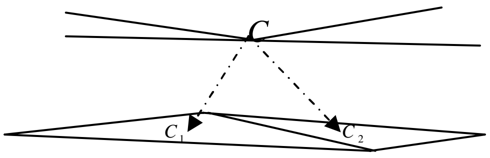


Fig. 4. C_i is defined as the center of the skin triangle, and then the direction from cloth vertex C to C_i is defined as the ray direction for ray-triangle intersection test

proper direction of the rays which has a significant impact on the accuracy of the collision detection method. Generally, there are three different methods to generate ray direction. The first method is set the normal direction of the cloth as the ray direction. It needs recalculating the normal of every cloth vertex for each frame and furthermore it can't deal with the situation that there is a drape in the collision area. The ray will intersect with a skin triangle in another section so the collision would be missed in this case. The second method takes the vertical direction from the cloth vertex to the corresponding bone as the ray direction, which is unsuitable for some areas such as shoulder block. In this paper, the direction (Fig.4) from the cloth vertex to the center of each skin triangle in the section is calculated as the ray direction. This method is easy to implement and has no collisions missed. The ray-triangle intersection test algorithm is discussed in [21].

3 Performances

Our algorithm has been implemented on a PC running Windows XP with a 2.2 GHz Pentium Ⅳ CPU, an NVIDIA GeForce 5200 GPU, and 512M of Main Memory. We

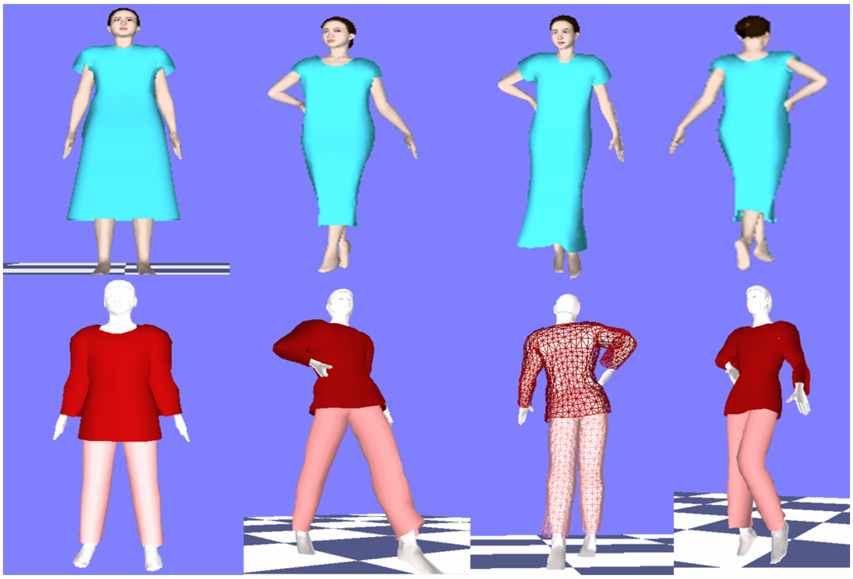


Fig. 5. a sequence of cloth animation (Cloth: 8,000 – 10,000 Tir ; Human: 7,500 - 9,000 tri ;Time : 50-100msec)

use it in cloth simulation and a virtual try-on system which will soon be introduced. We highlight the results on two cloth simulations shown in Fig.5. The number of triangles in the mesh used to model the body and clothes vary from 8k to 10k. The time to check for cloth-collisions is in the range of 50-100msec. The performance depends on the input complexity. During our implementation, the skin triangles are classified into thirteen joint blocks. Each block is divided into four segments and each segment is further divided into eight to ten sections. So there are only twenty skin triangles or so in a section. As a result, our algorithm obtains speedup due to no Skin-H updating operation and less number of elementary tests between the primitives.

Our approach has demonstrated that simulating dressed virtual human with middle-range PC is feasible. It can deal with several kinds of clothes such as skirts and suits. For further accelerate the method, some inactive cloth areas such as the areas near shoulder and waist can be bound to the corresponding joints in order to reduce the number of the collision tests. However, no self-collision is calculated in this paper, therefore, it can’t simulate the wrinkles well.

4 Conclusions

To our knowledge, it’s the first time that the generalized cross-sectional contour technique[10,18] is used for cloth-body collision detection. The proposed method has the following advantages.

Efficiency. The method can achieve near-real-time because most of the time-consuming operation was done during preprocessing step and the hierarchical update was avoided.

Accuracy. The method belongs to the hierarchical ones. The advantage of accuracy of the Hierarchical methods is inherited in a natural way.

Practicality. The method is very simple. It is easy to implement and can be applied to virtual modeling, virtual human animation and other applications.

Acknowledgements

This work is supported in part by NSFC-60403042, NSFC-60573162, NSFC-60533070, NSFC-60603082, CISTC-2005DFA11060, BNSFC-4051004, BNSFC-4062032, HTRDPC-2006AA01Z336.

References

1. Benes, B., Villanueva, N.G.: GI-COLLIDE: Collision Detection with Geometry Images. In: SCCG’05:Proc. of the Sping Conference on Computer Graphics, pp. 291–312 (2005)
2. Cordier, F., Magnenat-Thalmann, N.: Real-time Animation of Dressed Virtual Humans. Eurographics, Blackwell publishers, 21(3), 327–336 (2002)
3. Cordier, F., Hyewon, S., Magnenat-Thalmann, N.: Made-to-Measure Technologies for an Online Clothing Store. IEEE CG&A 23, 38–46 (2003)

4. James, D.L., Pai, D.K.: BD-Tree: Output-Sensitive Collision Detection for Reduced Deformable Models. ACM Trans. Graphics 23(3) (2004)
5. Fan, Z.W., Wan, H.G., Gao, S.: Streaming Real Time Collision Detection Using Programmable Graphics Hardware. Journal of Software(in Chinese with English abstract). 15(10), 1505–1514 (2004)
6. Baciu, G., Wong, W., Sun, H.: RECODE: An Image-based Collision Detection Algorithm. In: Proceedings of the 6th Pacific Conference on Computer Graphics and Applications, p.125, October 26-29, 1998 (1998)
7. Baciu, G., Wong, W.: Hardware-Assisted Self-Collision for Deformable Surfaces. In: Proc. ACM Symp. Virtual Reality Software and Technology, pp. 129–136, November 2002 (2002)
8. Baciu, G., Wong, W.: Image-Based Collision Detection for Deformable Cloth Models. IEEE Trans.Visualization and Computer Graphics 10(6), 649–663 (2004)
9. Chittaro, L., Corvaglia, D.: 3D Virtual Clothing: from Garment Design to Web3D Visualization and Simulation. In: Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology, pp. 73–84. ACM Press, New York (March 2003)
10. Yan, L., Zhaoqi, W., Tianlu, M.: A Method of Virtual Human Skin Deformation based on Generalized Cross-Sectional Contour. Computer Science(in Chinese with English abstract) 32(1), 190–193 (1) (2005)
11. Mezger, J.S., Kimmerle, O.: Etzmus. Hierarchical Techniques in Collision Detection for Cloth Animation. Journal of WSCG, 11(1) (2003)
12. Lin, M., Gottschalk, S.: Collision Detection between Geometric Models: A Survey. In: Proc. of IMA Conference on Mathematics of Surfaces 1998 (1998)
13. Teschner, M., Kimmerle, S., Zachmann, B., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., Volino, P.: Collision Detection for Deformable Objects. Eurographics State-of-the-Art Report, pp. 119–139 (2004)
14. Naga, K., Govindaraju, M.: Lin and Dinesh Manocha. Fast and Reliable Collision Detection Using Graphics Processors. In: Symposium on Computational Geometry, pp. 384–385 (2005)
15. Naga, K., Govindaraju, D., Knott, N., Jain, I., Kabul, R., Tamstorf, R., Gayle, M.C.: Interactive Collision Detection between deformable Models Using Chromatic Decomposition. ACM Trans Graphics 24(3), 991–999 (2005)
16. Volino, P., Magnenat-Thalmann, N.: Resolving Surfaces Collisions through Intersection Contour Minimization. In: ACM Transactions on Graphics (SIGGRAPH 2006 proceedings), vol. 25(3), pp. 1154–1159. ACM Press, New York (2006)
17. Rodriguez, J., Sainz, M., Susin, A.: Fast Body-Cloth Simulation with Moving Humanoids. Short presentations EG'05, pp. 85–88 (2005)
18. Jianhua, S., Magnenat-Thalmann, N., Thalmann, D.: Human Skin Deformation from Cross Sections. In: Proceedings of Computer Graphics International CGI'94 (1994)
19. Gottschalk, S., Lin, M.C., Manocha, D.: OBBTree: A Hierarchical Structure for Rapid Interference Detection. In: Computer Graphics (SIGGRAPH'96), pp. 171–180, ACM (August 1996)
20. Vassilev, T., Spanlang, B., Chrysanthou, Y.: Fast Cloth Animation on Walking Avatars. In: Computer Graphics Forum Proc. Of Eurographics (2001)
21. Moller, T., Trumbore, B.: Fast, Minimum Storage Ray-Triangle Intersection. Journal of Graphics Tools 2(1), 21–28 (1997)