A Study on Force Feedback Presentation System for Local Domain Based on Distributed Collision Detection

Tetsuya Takahashi Kenta Wada* Koichi Konno* Junji Sone** Yoshimasa Tokuyama***

*Graduate School of Engineering, Iwate University
**Faculty of Engineering, Tokyo Polytechnic University
E-mail: t-tetsuya@lk.cis.iwate-u.ac.jp

1. Introduction

Constructing virtual reality (VR) system has become easy and possible since low-cost and high-performance hardware becomes popular. The VR technology is expected for various kinds of application such as virtual museum, medical engineering, education, video games and other graphic activities. A lot of element technologies to construct such VR environments have been studied, while integrating the element technologies still costs much. This problem arises because response speed of ordinary computers is very slow and expensive computers must be prepared to integrate the element technologies.

One of the solutions about the cost and the processing speed is to decentralize element technologies. We build a VR system to disperse a large amount of calculation to a PC cluster and to integrate the calculation results. In addition, the client PC generates multiple threads by using thread programming. By dispersing the processing loops into two threads, the force feedback presentation thread and the distributed collision detection thread, the problem of response speed will be solved.

In this paper, we have studied the virtual touching system that uses the collision detection with a PC cluster. Our system realizes high-speed collision with a complex object that has a large number of polygons and presents force feedback in a local domain of collision. Using a PC cluster to calculate collision improves response speed. In addition, we construct a system to present smooth force feedback by generating artificial force feedback in the force feedback presentation thread, where the amount of calculation is small and processing speed is high.

2. Related Work

The methods of conventional distributed collision detection have been studied by using bounding box or Octree techniques to achieve real-time processing [3]. There is a problem, however, that precise collision detection takes long time if it is performed in a general PC. The method [4] to speed up the collision detection with Graphics Processing Unit (GPU) is suggested, although this method [4] has a problem that the number of polygons in a model is limited because of the amount of memory in the GPU. To solve these problems, the method [5] is suggested that uses a PC cluster where multiple general PCs are connected, so that a polygon model with a huge number of polygons can be processed. The PC cluster is utilized widely in this way, but there are few practical VR systems that integrate the collision detection and the force feedback presentation. B. Raffin et al. proposed that a PC cluster could shorten the calculation time of high-cost processing such as collision detection [6]. Their technique needs a special library since it is realized in UNIX environment and MPI [7] is assumed for distributed processing.

On the other hand, in the Windows environment, DCOM (Distributed Component Object Model) [8] is offered for distributed processing. DCOM is the specifications of the dispersion agenda technology that Microsoft Corporation established to exchange data or to communicate for processing requests in a network. Since DCOM does not use special software on the Windows OS, a computer environment can be established easily. In our study, the DCOM is most suitable because idle PCs are used in a cluster.

For the force feedback presentation technique, HAPTEX system [9] that integrates sense of touch, haptic device and sense of sight is studied broadly at present; for example, precise force feedback presentation with tolerance of several grams, haptic device operation, development of new algorithms about haptic information processing and development of new haptic devices.

The recent algorithms adopt the structures such as multi-layer or multi-thread measures to achieve real-time processing [9], although none of the algorithms suggests concrete system configuration.

3. Proposal of Virtual Touching System

3.1 Hardware configuration

To realize a virtual touching system described in this paper, a client-server model shown in Figure 1 must be constructed. For distributed processing of the huge amount of calculation in real time, we use the cluster system where multiple PCs are connected in gigabit network. In addition, a client PC is connected to the same

gigabit network. To the client PC, a haptic device is connected to present force feedback. In our study, eight computers are connected to construct a PC cluster. The CPU of each computer is Pentium4 3.0 GHz with 1.0 Gbyte RAM, connected in gigabit network. As a haptic device, PHANToM Omni is connected to the client PC.

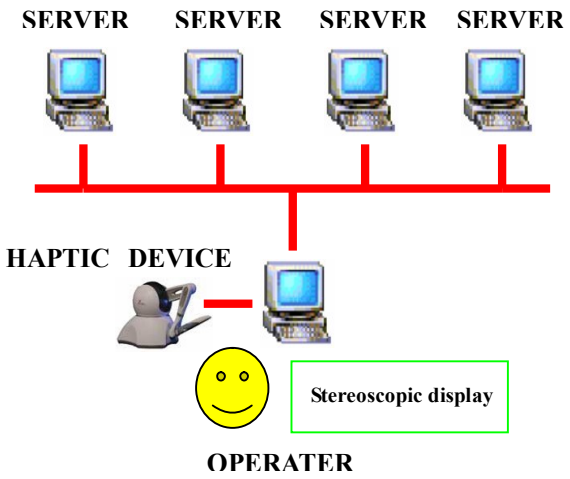


Fig. 1: Client-server model

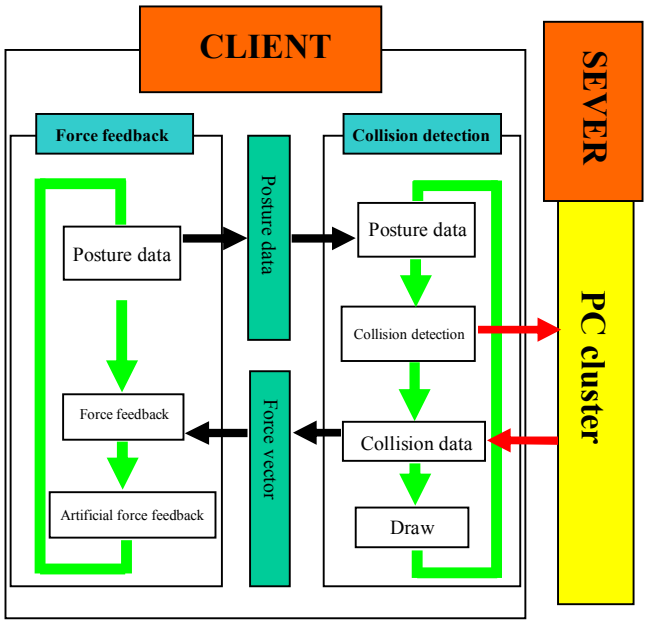


Fig. 2: Software configuration

3.2 Software configuration

PHANToM Omni enables high-speed processing of 1 KHz I/O control update rate. In our study, in order to maintain responses of force feedback presentation, multiple threads are generated through thread programming and the processing loops are divided. Figure 2 shows the outline of the thread processing in the client PC. In the client PC, two threads are constructed: one for force feedback presentation and the other for dispersion collision detection. The thread for dispersion collision detection performs independent calculation processes. Details of the processing in each of the threads are described section 3.6.

3.3 Collision detection method using layered boundary spheres

In our study, collision detection method based on layered boundary spheres [5] Fujiwara suggested for real time system construction is used. This method, based on the concept for which Octree is expanded, generates layered boundary spheres whose number of faces in the sphere of the lowest layer will be almost constant.

In Fujiwara's method, a PC cluster is used to shorten the calculation time for collision. As the collision detection method using the layered boundary spheres is distributed in multiple computers, calculation cost of each computer can be reduced and the faster processing can be realized. Details of the technique of how to distribute layered boundary spheres to a PC cluster are described in section 3.4.

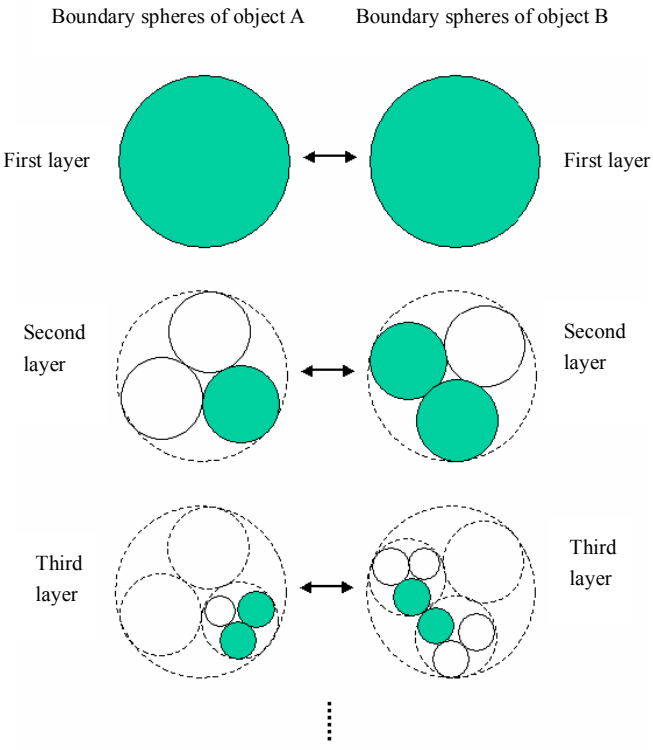
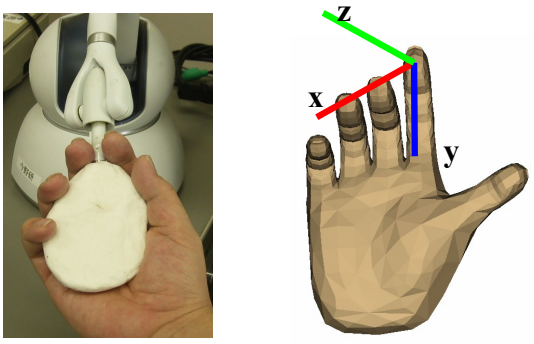


Fig. 3: Procedure of collision detection

3.4 Outline of distribution technique

"Block-cyclic mapping [10]" is one of the distribution techniques, which can allocate spheres of the ***n***th layer of object A of Figure 3 to each computer evenly and discretely. In the block-cyclic mapping, a partial space where a shape exists is allocated to two or more computers, so that calculation load is balanced in every computer. Each

computer detects colliding faces about the allocated spheres and the client computer integrates the calculation results of every computer in the end, so that all collisions are detected.



(a) (b)

Fig. 4: Appliance attached to PHANToM Omni and a hand model

3.5 Evaluation of distribution technique

In our study, as Figure 4 shows, the tip of a finger is linked with the tip of PHANToM pen, so that a system is constructed in a virtual space where the pen tip of PHANToM Omni moves synchronously with the hand model. As an operator uses PHANToM Omni to move the hand model, the client computer detects collisions between the hand and an object while cooperating with the servers. The client computer then integrates the collision detection results sent from the servers and displays the integrated results.

Colliding faces of the hand and the horse are displayed in red to clarify collision faces (Figure 5). In this paper, the total number of colliding faces of the object and those of the hand is called "the number of contacts [11]". The number of contacts largely influences the time length of collision detection. The number of contacts depends on the number of polygons of the hand model and the object or the contact area. As the number of contacts increases, calculations for displaying the sense of force will be more precise and the reality of display polygons will increase. Then, with variable number of contacts, the time length for collision detection is measured and the difference of responses derived from variable numbers (1, 2, 4, 8) of the servers is evaluated. This evaluation is the most important subject when a virtual touching environment is constructed.

In our study, about a hand model (34,432 polygons) and a horse model (96,966 polygons), we evaluated response time length while changing the number of the servers (0, 1, 2, 4 and 8) of the PC cluster.

Table1: Number of contacts between a hand and a horse (compared to the case with no server) (in sec.)

Contacts	**0**	**1**	**2**	**4**	**8**
100	0.18	0.25(0.7)	0.23(0.8)	0.31(0.6)	0.53(0.3)
200	0.33	0.32(1.0)	0.27(1.2)	0.33(1.0)	0.55(0.6)
300	0.52	0.33(1.6)	0.37(1.4)	0.36(1.4)	0.59(0.9)
400	0.69	0.54(1.3)	0.47(1.5)	0.42(1.6)	0.64(1.1)
500	1.02	0.64(1.6)	0.44(2.3)	0.43(2.4)	0.67(1.5)
1000	2.32	1.31(1.8)	1.03(2.3)	0.89(2.6)	0.75(3.0)

Table 1 shows the change of response time length when the number of contacts is about 1,000 or smaller in collision detection between the hand and the horse. The boldface numbers in the leftmost column of Table 1 are the number of the contacts. When the number of the contacts is about 100, the response time length increase almost linearly as the number of servers increases. This occurs because the affine transformation time will be longer than collision detection time if the number of PCs increases. As the number of contacts reaches almost 1,000, the decrease ratio of time for collision detection becomes greater than the time length for affine transformation, so that increasing the number of servers decreases the response time length. Therefore, when the number of contacts is smaller than 1,000, one or two servers are suitable for efficient processing. In Table 1, note that the values in the parentheses indicate comparison of processing speed to that with 0 server.

3.6 Force feedback presentation method and thread processing

Generally, in order to contact with an object in a virtual environment, it is necessary to specify a contact spot and to calculate the size and the direction of force on the contact spot. In this paper, the contact spot between the hand model and the object is obtained by using the technique described in sections 3.3 and 3.4.

For the force feedback generation with PHANToM Omni, it is necessary to generate force feedback vectors to express the size and the direction of the force on a most suitable point. Figure 6 (a) shows the method to generate the force feedback vector implemented in our study. Pink circle of Figure 6(a) shows contact face group between the hand model and the object. The distances between the vertices of all contact faces of the hand model and the centers of gravity of all contact faces of the object are calculated so that the face of the object model where the distance is the shortest is obtained. Figure 6 shows the face of the shortest distance with a square marker. In our

technique, the normal vector of the face belonging to the object is assumed to be the direction of the force feedback vector. As Figure 6 (b) shows, as a hand model contacts a face of the object, and the size of the force feedback vector generates force in proportion to the distance (penetration distance) that the hand model moves to the object further. To be concrete, the hand model records the collision face that contacts with the object first and generates force feedback according to the face where the shortest distance is obtained and the penetration distance recorded as the distance with the collision face.

The equation to perform the force feedback presentation is expressed by the following expression:

$$F = ((\log(PD) + a) * b) * FD \qquad (1)$$

where F is force to be feedback, PD is the penetration distance between collision face and face of shortest distance, FD is the force feedback vector provided from a collision face, and a and b are constants by the experience value, value of a is 3.0, value of b is 0.8.

A logarithmic function is available to obtain force feedback due to touching an object more definitely than force feedback presentation by using a simple spring model.

If two or more collision faces are obtained, the method mentioned above can generate force feedback vectors. Figure 5 shows an example of touching simulation with a hand model and a horse model. The contact faces between the hand model and the horse model are displayed in red, which indicates that collision occurs on multiple spots. Since the I/O control update rate PHANToM Omni is 1 KHz, force feedback is not generated continuously if calculation time of collision detection is long. This is the problem that it is difficult to generate the continuous and smooth force feedback.

Then, as described in section 3.2, the client PC generates two threads, force feedback presentation and distributed collision detection, so that the force feedback presentation continues during collision detection. The consistency of the force feedback presentation can be maintained by separating a processing loop. Figure 2 shows the outline of the configuration of the processing. The thread of force feedback presentation performs the processing to show force feedback in PHANToM Omni, and the hand posture data of Proxy provided from PHANToM Omni is sent to the dispersion collision detection thread. Based on the posture data, the dispersion collision detection thread detects collision by using the PC cluster. From the collision face data obtained through the collision detection, a force feedback vector is generated with using the method mentioned earlier and the vector is sent to the force feedback presentation thread. The force feedback presentation thread, based on the force feedback vector, generates force feedback of expression (1). Since writing the posture data and the force feedback vector shown in Figure 2 performs exclusive control, the force feedback vector cannot be obtained during data writing.

In addition, since the processing of collision detection thread needs almost 25 times longer time than that of force feedback presentation thread, an artificial force feedback vector is generated until the next data is obtained.

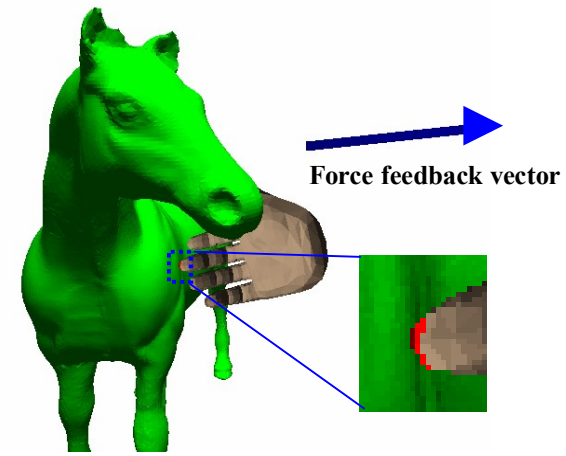


Fig. 5: Sample touching simulation between a hand model and a horse model

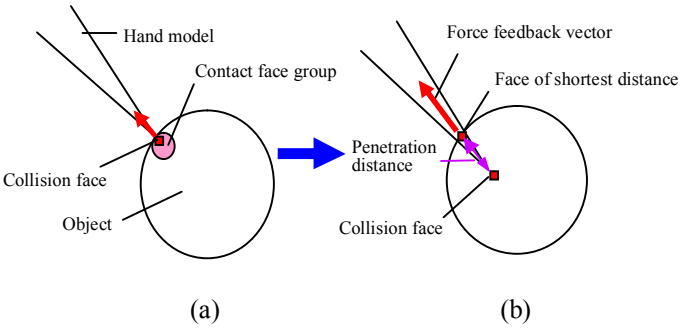


Fig. 6: Generation of force feedback vector

3.7 Artificial force feedback generation

The artificial force feedback vector suggested in our study is generated based on the amount of movement of the hand model and the force feedback vector before the movement. In our technique, the force feedback vector after the movement is estimated from the amount of movement of the hand model. By adding the vector of the movement to the force feedback vector before movement, an artificial force feedback is generated to interpolate the force feedback vector before the movement and the one after the movement. As shown in Figure 7, vector D is added to force feedback vector A before movement of the hand model, so that artificial force feedback vector B is generated to interpolate vector A and

force feedback vector C after the movement.

Figure 8 shows the force feedback vectors of Figure 7, which can present smooth force feedback.

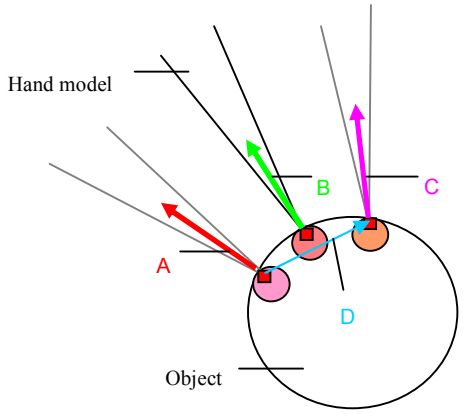


Fig. 7: Artificial force feedback

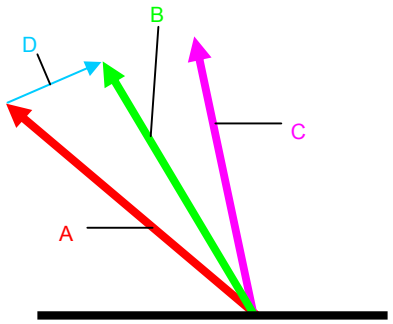


Fig. 8: Artificial force feedback vector

4. Results and Consideration

Our study aims at real-time virtual touching simulation. On this viewpoint, although the number of the contacts varies according to objects with the different numbers of the polygons, it is expected that processing must be as close as real time. Therefore, in order to clarify the most suitable number of servers for a PC cluster for different number of polygons and contacts, the four models are examined: the hand model of Figure 4, the horse model of Figure 5, a boat model, and a model subdivided by Loop subdivision method [12] so that the number of the polygons will be four times greater than the polygons of the boat. The hand model has 37,248 polygons, the horse model has 96,966 polygons, a boat model has 194,092 polygons and a subdivided model has 776,368 polygons. About these four models, the time length of force feedback presentation in local area is calculated based on dispersion collision detection method. In the experiment, two kinds of callbacks are counted: one is HL FPS that represents the number of times of force feedback presentation callbacks in one second and the other is GL FPS that represents the number of times of drawing callbacks in one second. The result is compared with the sample program attached to Open Haptics toolkit where a similar technique is provided.

The time length is calculated with different numbers of servers (0, 1, 2, 4 and 8) and the difference of the response from the sample program is evaluated. The results are shown in Tables 2 and 3. As Table 2 shows, HL FPS and GL FPS decrease as the number of polygons increases in the sample program. The reason of this result is assumed that only one thread performs force feedback presentation, interference calculation and drawing with using haptic library. If collision detection or drawing takes long time due to a great number of polygons, force feedback presentation would take long time accordingly. Moreover, for an object having a great number of polygons, it is hard to present real-time force feedback. If an object has 200,000 or more polygons, force feedback is hardly presented. On the other hand, with our technique, as shown in Table 2, although the number of polygons increases, frame rate for collision detection does not affect so seriously to force feedback presentation or drawing, since the processing of force feedback presentation, collision detection and drawing are separated into each processing loop.

Table2: Result of comparison between a sample program and the technique suggested (FPS)

	Sample		Technique suggested	
FPS	HL	GL	HL	GL
Ball (960)	780.54	68.38	1000	92.91
Horse (96,966)	6.21	45.4	1000	24.39
Boat (194,092)	2.74	16.83	1000	9.62
Boat (776,368)	1.38	6.33	1000	2.65

Table3: Increase of the number of servers and the number of contacts (Interference calculation FPS)

	Ball (960)		Horse (96,966)		Boat (194,092)		Boat (776,368)	
Contacts	100	1500	100	1500	100	1500	100	1500
0	2.25	0.12	2.19	0.09	1.29	0.08	0.99	0.07
1	4.65	0.26	2.26	0.11	2.33	0.13	1.55	0.19
2	4.78	0.43	1.82	0.22	2.74	0.21	1.72	0.22
4	2.44	0.69	3.34	0.37	1.83	0.66	1.66	0.43
8	1.62	0.42	1.78	0.71	1.67	0.57	1.43	0.46

From these results, it is thought that our technique can be applied to force feedback presentation for an object having the huge number of polygons. The response of collision detection with using a PC cluster is described in Section 3.5. When the number of contacts (total number of collision faces between a hand model and an object) is smaller than 1000, one or two servers are the most efficient to present force feedback; while for 1,000 or more contacts, the more the servers are used, the more effective the force feedback is presented. A similar conclusion can be found in the experiment result shown in Table 3.

5. Conclusion and Future Works

In this paper, the method to construct a virtual touching system based on the distributed collision detection by using a PC cluster is described. In our technique, the force feedback presentation, collision detection and drawing are performed in separate threads. Separating collision detection of force feedback presentation from the force feedback presentation loop maintains consistency of force feedback presentation, except time lag of the force feedback presentation. Generating artificial force feedback in the force feedback presentation thread can present smooth force feedback presentation. Therefore, with our technique, force feedback can be presented for an object having huge number of polygons, for which the sample program of Open Haptics toolkit cannot present force feedback. In addition, distributing collision detection to computers in a PC cluster reduces calculation time of each computer, which can improve entire performance.

Acknowledgement

This work was partially supported by KAKENHI (20500880).

REFERENCES

[1] Yoshifumi Kitamura, Narendra Ahuja, Haruo Takemura, and Fumio Kishino, “Colliding Face Detection among 3-D Objects using Octree and Polyhedral Shape Representation”, The journal of JRSJ, Vol.14, No.5, pp121-130, 1996

[2] Tetsuya Yokoyama, “Soft Object by Finite Element Method with Condition of Constraint in Contact Point”, The Journal of VRSJ, Vol.10, No.2, pp241-248, 2005

[3] Kenichi Kobori, Daisuke Nakanishi, “A Fast Collision Detection Using Filled Spheres”, The journal of IEICE Vol.J85, No.9, pp.1455-1463, September 2002.

[4] Yoo-Joo Choi, Young J. Kim, Myoung-Hee Kim, “Rapid Pairwise Intersection Tests Using Programmable GPUs”, The Visual Computer, Vol. 22, No.2 , pp. 80-89, 2006

[5] Shinya Fujiwara, Koichi Konno, Junji Sone, Yoshimasa Tokuyama, “A Method for Exact Collision Detection of Faces with Layered Boundary Spheres”, The Journal of IIEEJ Vol.35, No.1, pp. 20-29, 2006

[6] B. Raffin and L. Soares, “PC Clusters for Virtual Reality”, IEEE Virtual Reality Conference, 2006

[7] W. Gropp, E. Lusk, and A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, Scientific and Engineering Computation Series, The MIT Press, 1994

[8] FRANK E. REMOND III, DCOM development guide, Shoeisha Publishers, 1998

[9] M. Bergamasco and F. Salsedo et. al, “High Performance Haptic Device for Force Rendering in Textile Exploration”, The Visual Computer, Vol. 23, No. 4, pp. 247-256, 2007

[10] Peter S. Pacheco, Parallel Programming with MPI, Baifukan Publishers, 2001

[11] K. Wada, K. Konno, J. Sone, Y. Tokuyama, “An Investigation of Calculation Performance to Construct a VR System Based on Distributed Collision Detection”, Proceedings International Workshop on Advanced Image Technology, pp. 153-158, 2007

[12] C. T. Loop, “Smooth Subdivision Surfaces Based on Triangles”, Master’s thesis, University of Utah, 1987