TELKOMNIKA, Vol. 11, No. 6, June 2013, pp. 3115 ~ 3122
e-ISSN: 2087-278X

Hybrid Collision Culling by Bounding Volumes Manipulation in Massive Rigid Body Simulation

Norhaida Mohd Suaib*[1], Abdullah Bade[2], Dzulkifli Mohamad[3]
[1,3]Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia.
[2]School of Science & Technology, Universiti Malaysia Sabah, Kota Kinabalu, Sabah, Malaysia.
Corresponding author, e-mail: haida@utm.my, abade08@yahoo.com, dzulkifli@utm.my

Abstract

Collision detection is an important aspect in many real-time simulation environments. Due to its nature of high Computation involved, collision detection can contribute to the bottleneck on the system involving large number of interacting objects. This paper focuses on finding options to efficiently cull away object pairs that are not likely to collide in large-scale dynamic rigid-body simulations involving n-body objects. The main idea is to perform time critical computing concept by manipulation of potential bounding volume techniques. In order to take advantage of a fast collision test and a more accurate result, a hybrid collision culling approach based on sphere-or-Dops was used. Based on initial results, this approach shows a potential adaptation to a massive rigid body simulation.

***Keywords**: collision culling, collision detection, rigid body, bounding volume, boundary representation*

1. Introduction

Collision detection is usually an integral part to many real-time and interactive applications. There are many applications that relies on collision detection such as in the field of virtual reality, computer games, animation, training, engineering and medical simulation. With these vast areas of collision detection applications, it is understandable that collision detection research is still an active research area. Despite the advances in technology, collision detection still contributes to the system bottlenecks for more than thirty years [1].

In many real-time and interactive applications, collision detection is a pre-requisite for realistic system response. A simple example is involving a computer generated bouncing ball. Collision between the ball and any obstacles need to be detected so that the system can generate appropriate response like the direction and momentum after impact. Collision detection and collision response usually are carried out successively [2].

The main purpose of collision detection is to ensure that there will not be any two objects that occupy the same space at the same time. This is equivalent to how objects behave in the real world–two objects cannot fit into the same space at exactly the same time. Depending on the level of accuracy that is needed on the application, collision detection process can either simply flag intersection(s) between objects, or it might produce detailed report on the event like time of contact, point of contact and interpenetration depth. The first type of detection can be carried out in a very short time but does not yield much information about the collision. Application like computer games that requires approximate but fast collision detection usually adopts these types of collision detection approaches. On the other hand, more computation needs to be carried out in order to get detailed collision information needed for a more serious application as in medical simulator. Therefore there is generally trade-offs between speed and accuracy in collision detection. At the same time, issues like real-time performance, efficiency and robustness need to be addressed [1].

Perhaps the most adapted approach in trading speed and accuracy was introduced by Hubbard as can be seen in many research papers concerning collision detection. It works based on the idea of time-critical computing: collision detection and response for interactive graphics applications can be improved by using a two-phase process: broad-phase and narrow-phase [3]. Collision culling, which in essence deals with quickly removing object pairs that are

Received January 18, 2013; Revised March 30, 2013; Accepted April 16, 2013

less likely to collide is very much related to the broad-phase level, as will be discussed thoroughly in the next section. This will be the focus of this paper.

The rest of this paper will be organized as follows: related work, particularly involving bounding volume in broad-phase collision culling will be discussed in Section II followed by the idea of hybrid collision culling in Section III. Experimental layout will be outlined in Section IV. Section V deals with the result and discussion, while Section VI concludes this paper.

2. Related Work

As mentioned in previous section, a two-phase approach is usually adopted in minimizing collision tests due to the nature of collision detection that is computationally intensive. Generally, the broad-phase collision detection acts more like a filter that identifies only pairs of objects that are more likely to collide. This step is responsible to cull away unrelated pairs in the whole collision detection process, and thus synonymous with the term ‘collision culling’. These identified pairs are then fed to the narrow-phase collision detection for further collision tests (please refer to [4] for an elaborate discussion on collision detection approaches and applications).

Different approaches can be implemented during collision culling process such as the brute force (all-pair test), sweep and prune (SaP) and hierarchical hash table [4]. Method implemented in this study is based on bounding volume technique which is very conventional with the first approach. Conventional bounding volumes that are commonly used in the broad-phase level will be presented next.

2.1. Conventional Bounding Volumes

Bounding volume is one of the most commonly used broad-phase collision detection approach in simulations involving n-body objects. Based on its popularity, we will next outline some of the popular bounding volumes.

Bounding volume algorithms encompasses techniques like bounding sphere, Axis-aligned bounding box (AABB), oriented bounding box (OBB) and discrete orientation polytopes (k-DOPs) (see Figure 1). Oriented-Dops (or-Dops), a combination of OBB and k-Dops was introduced to overcome the update cost of k-Dops [5] (shown as Figure 2).

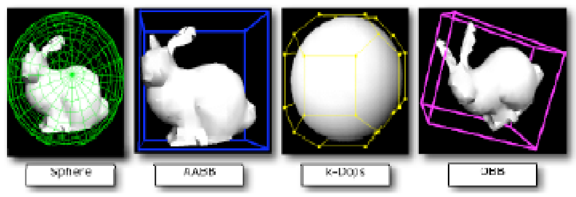


Figure 1. Conventional Bounding Volumes

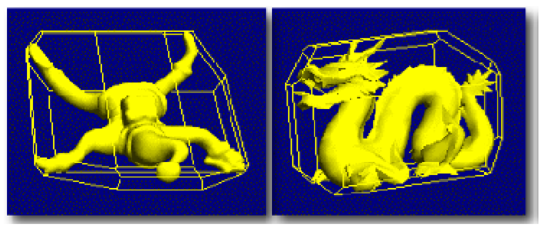


Figure 2. Oriented-Dops or also known as or-Dops

2.2. Bounding Volumes: Simplicity Versus Accuracy

Based on previous research, these bounding volumes had its own advantages and disadvantages. In short, a simple bounding volume requires less computation in terms of construction, updates and tests. Therefore it is expected that simulations or applications utilizing simple bounding volumes can give higher frame rates but with the cost of more false positive test outcome. A false positive collision occurred when collision test based on bounding volume gives a positive result (collision detected) but the actual object did not collide. This is due to large empty corners caused by simple bounding volume. On the other hand, a more accurate bounding volume requires more computation and more time, resulting in less frame rates. The previous bounding volumes can be roughly arranged according to simplicity versus accuracy as in Figure 3.

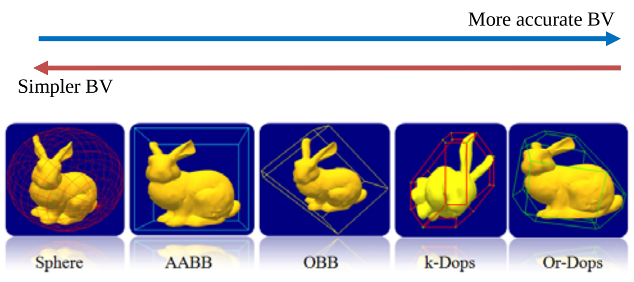


Figure 3. Simplicity versus Accuracy

2.3. Time-Critical Collision Detection

The idea of time-critical collision detection, as introduced by Hubbard [3], has been adapted by many researchers. Although it is quite impossible to include all references in this category, some of highly related work will be discussed here. Most of the approaches use bounding volume hierarchy (BVH) in one way or another. One of them utilized the tree called average-distribution tree or ADB-tree [6] that is a combination of bounding volume hierarchy (BVH) with an estimated probability of collision occurrence to reduce collision tests. Sphere-tree was used for time-critical collision detection by at least two researchers; sphere-tree on reduced model for deformable objects [7] and sphere-tree with closest feature maps (CFMs) applied to refinable collision response [8, 9]. An AABB-tree with reduced deformable model was also used for self-collision culling [10]. An event-based scheduling that adaptively prioritizing collision tests and performs collision tests at different time interval was introduced by Coming and Staadt [11, 12], while a BVH was used with certificates that indicate absence of self-collision [10].

2.4. Total Cost Benchmarking

Total cost benchmarking (Equation 1) for collision detection that was proposed by [13] will be used as a basis:

$$T = Nu \times Cu + Nv \times Cv + Np \times Cp + Co \quad (1)$$

Where:

T: total cost function for interference detection,
Nv: number of bounding volume pair overlap tests
Cv: cost of testing a pair of bounding volumes for overlap
Np: is the number primitive pairs tested for interference,
Cp: cost of testing a pair of primitives for interference,
Nu: number of bounding volumes updated,
Cu: cost of average bounding volume update,
Co: indicates cost for one time processing, where necessary

However, not all parameters must be involved; they depend on the problem and the type of collision involved.

3. Hybrid Collision Culling Method

The main purpose of collision culling is to reduce collision tests by identifying and culling away unnecessary pairs. The basic collision detection process which usually is carried out in a two-phase process (broad-phase and narrow-phase) inspires a two-phase collision culling process in order to achieve a simple, fast and reliable collision detection.

The culling process will be based on the bounding volume approach. Since there are quite a number of popular bounding volume, a question will needs to be address is the type of bounding volumes that is suitable for that purpose. If there are suitable candidates, is there any way that performance of the culling process can be improved?

There are two main issues that need to be considered in the question above. On the one hand, a simple and fast collision test is needed but on the other hand, it also needs to be reliable. The candidates for simple and fast collision test are sphere and AABB, mainly due to their simple representation and tests. However, it is typical for these ‘simple’ bounding volume to lead to false collision detection due to large empty corners. Therefore, it is desirable to look into the possibility of manipulating a more accurate bounding volume so that it could reduce the false detection.

4. Experimental Layout

For the moment we are experimenting on different BVs to be used on the avatar. At this stage, we are targeting a suitable BV to be used on many avatars and probably very close to one another, like in a crowded environment. Therefore, a BV that fits an avatar (meaning that the empty space is not too large) but could support real-time application is very much preferred.

Since movement of VE inhabitants could not be anticipated most of the time, we design a set of simple experiments where we test four different types of BVs (sphere, AABB, k-Dops and OBB) which are most commonly found in the literature, plus another type of BV called oriented Dops (Or-Dops). At this stage, a single bounding volume was assigned for each object, and each object was assumed to represent an avatar.

4.1. Hardware Used

All of the experiments outlined here were done on a CPU running on an Intel® Core™2 Duo CPU E7400 @2.80GHz processor with 2 GB RAM onboard, and NVidia GeForce 210 with Microsoft Windows XP Professional SP3. Different types of bounding volumes were tested using the same hardware configuration as will be explained in the next sub-section.

4.2. Experiments

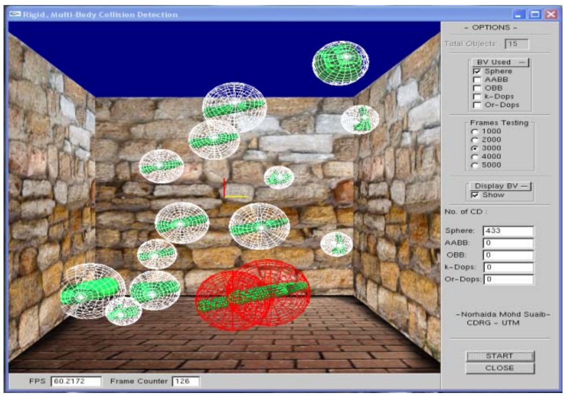
There are mainly two experiments involved: the first part was designed to identify the suitable bounding volumes that could be used for the hybrid collision culling, and the second part was to experiment on the performance of the proposed hybrid method.

4.2.1. Part I

Four sets of experiments were involved in the first part. All four experiments involved the commonly used bounding volumes – sphere, AABB, OBB, k-Dops and or-Dops that were used to bound 15 different objects. Objects used are based on point-cloud, ranging from 74 to 1346 vertices as listed in Table 1.

Table 1. List of 3D Objects Used

	List of 3D Objects Files	Total Vertex
1	[head.tri]	542
2	[torso.tri]	841
3	[pelvis.tri]	74
4	[upperRightLeg.tri]	706
5	[upperLeftLeg.tri]	706
6	[lowerRightLeg.tri]	706
7	[lowerLeftLeg.tri]	706
8	[leftFoot.tri]	122
9	[rightFoot.tri]	122
10	[upperLeftArm.tri]	1346
11	[upperRightArm.tri]	1346
12	[lowerLeftArm.tri]	1138
13	[lowerRightArm.tri]	1138
14	[leftHand.tri]	589
15	[RightHand.tri]	589



Total Objects: 15

BV Used: [✓ Sphere] AABB OBB k-Dops Or-Dops

Frames Testing: 1000 2000 [✓ 3000] 4000 5000

Display BV: [✓ Show]

No. of CD :

Sphere: 433
AABB: 0
OBB: 0
k-Dops: 0
Or-Dops: 0

-Norhaida Mohd Suaib-
CDRG - UTM

FPS 60.2172 Frame Counter 126

Figure 4. Sample experiment using bounding spheres

Average bounding volume construction times for these objects will be logged. For the next experiments, all objects are randomly transformed inside a volume (box) with different with different velocity assigned as in rigid body motion, so that there will be collisions among these objects. Figure 4 shows one of the experiments conducted on bounding spheres. Besides capturing the average construction time, there are three other experiments that were conducted.

The first experiment was conducted to test collision time using different types of bounding volumes while the second experiment was conducted to test the average update time. Results from these experiments were fed into the Total Cost Benchmarking function. The last experiment was implemented to roughly show the overall performance based on frames-per-second (FPS) counts.

4.2.2. Part II

The second part of this test deals with further experiments on the proposed hybrid collision culling method. As mentioned earlier, the hybrid collision culling is a two-phase collision culling process designed to achieve a simple, fast and reliable collision detection. The scene involves a combination of randomly moving objects and some static objects inside a box (see Figure 5a and 5b for a sample of loaded objects). Similar to experiments in Part I, objects are allowed to go through other objects but will simply change direction once they hit the boundary of the box to ensure that no object will wander off the boundaries. Once collision is detected, a change of colour will visually indicate collision between two objects and related information will be logged. While experiments in Part I only involved relatively small number of objects, the proposed method will be tested against conventional techniques in a massive rigid body simulation undergoing rigid-body simulation.

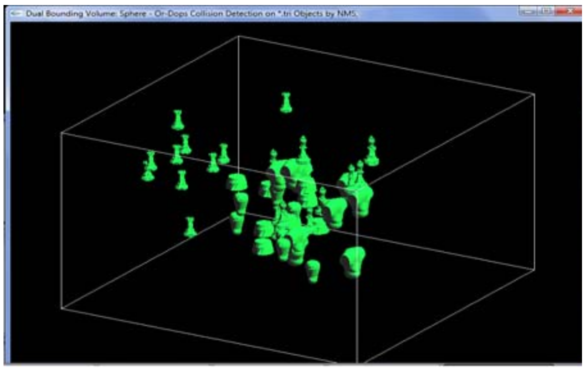

Dual Bounding Volume: Sphere - Or-Dops Collision Detection on *.tri Objects by NMS

Figure 5a. Sample Experiment using 50 Objects without BV

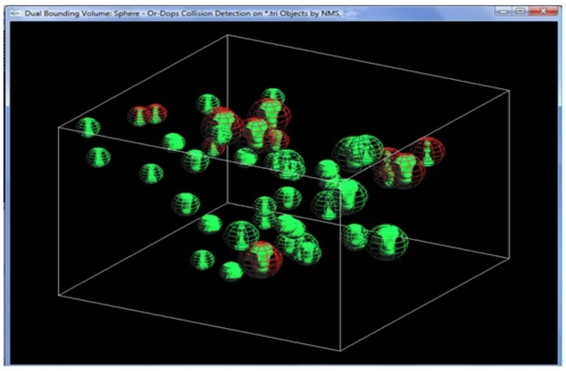

Dual Bounding Volume: Sphere - Or-Dops Collision Detection on *.tri Objects by NMS

Figure 5b. Sample Experiment using 50 Objects with Bounding Sphere

5. Results and Analysis

5.1. Part I

Construction Time. Construction times for each objects used (as previously listed in Table 1) was recorded and repeated 100 times. Average construction time is shown in Figure 6.

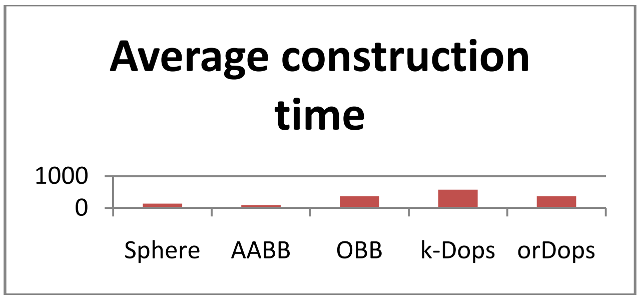


Figure 6. Effects of Selecting Different Switching under Dynamic Condition

Number of detected collisions. Initially, accumulated number of collisions detected was recorded for 100, 200, 300, 400 and 500 frames as indicated in Table 2 and Figure 7.

Table 2. Numbers of Collisions Detected

Bounding Volumes	100	200	Frames 300	400	500
Sphere	367	572	872	1091	1441
AABB	159	255	387	497	625
OBB	82	134	207	308	395
k-Dops	71	99	165	230	316
or-Dops	59	86	140	202	267

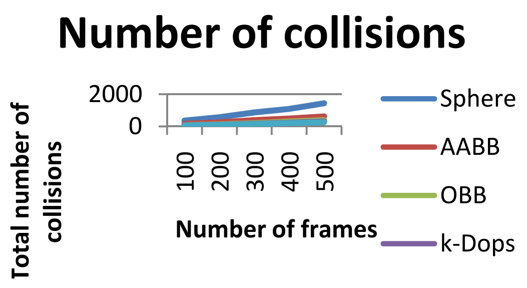


Figure 7. Number of collisions detected

Average frames per second (FPS). Although frames per second (FPS) is not a definite measure for performance, the fps values for different types of bounding volumes used were included for indicative purposes. Figure 8 illustrates the fps values.

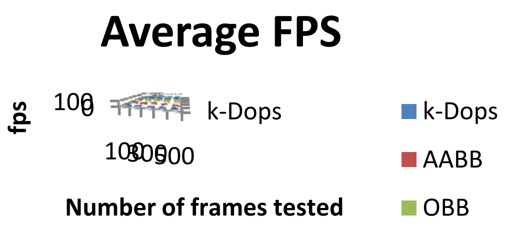


Figure 8. Frames per second for the first 500 frames

Total Cost Benchmarking. A modified benchmarking function based on Eq. 1 was used since collision tests conducted in these experiments do not involve primitive tests. Therefore, the values of Np and Cp were not included and total cost function only involve:

$$T = Nu \times Cu + Nv \times Cv + Co \quad (2)$$

Where

T: total cost function for interference detection,
Nv: number of bounding volume pair overlap tests
Cv: cost of testing a pair of bounding volumes (overlap)
Nu: number of bounding volumes updated,
Cu: cost of average bounding volume update,
Co: indicates cost for one time processing, in this case, the construction cost

Results from experiments for the first 500 frames are summarized in Table 3 and Figure 9 below. Calculations for total cost were based on Eq. 2 mentioned above.

Table 3. Numbers of Collisions Detected

BV	Values					Total Cost
	Nv	*Cv*	*Nu*	*Cu*	*Co*	
Sphere	52395	0.00024	104790	0.00012	135.0783	160.2126
AABB	52395	0.00024	104790	0.049377	90.63781	5277.364
OBB	52395	0.100106	104790	0.050053	367.6315	10857.72
k-Dops	52395	0.000249	104790	0.000124	574.2866	600.3288
or-Dops	52395	0.001567	104790	0.000784	369.4002	533.6356

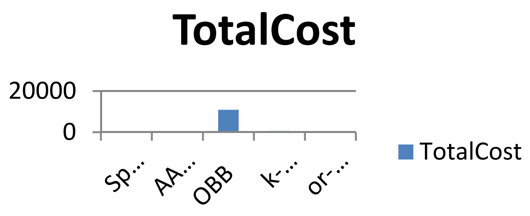
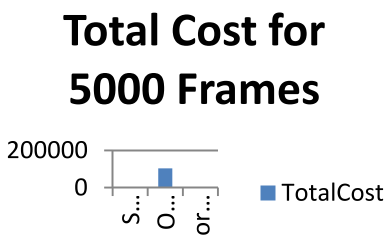


Figure 9. Total cost based on first 500 frames

The total cost benchmarking process was also carried out for 1000 and 5000 frames simulations. Experiments involving 1000 frames gave unexpected result where the value of total cost for k-Dops skyrocketed above OBB. Based on the first 500 frames tested, AABB and k-Dops gave frame rates way below interactive frame rates (~60 fps), and this might contribute to the lapse. These two bounding volumes are excluded in the 5000 frames test. Results for the final test (5000 frames) are as expected; sphere gave the lowest cost, followed by or-Dops (see Figure 10).



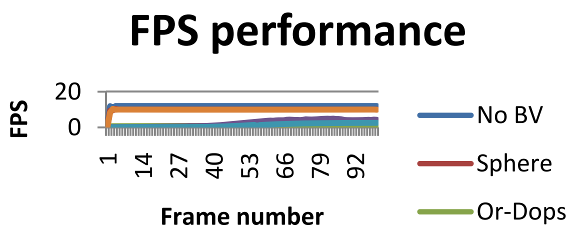
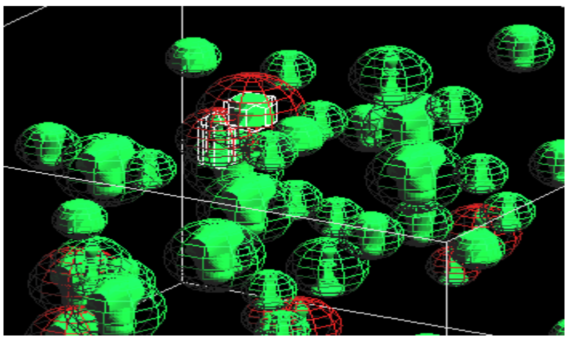


Figure 10. Total Cost based on Shortlisted BV and 5000 Frames

Figure 13. Fps Performances for Different BV Approach for 500 Objects (first 100 frames)

5.2. Part II

Hybrid Collision Culling. Results from the experiments in Part I show that combination of sphere and or-Dops bounding volumes offer the potential for an option towards a better collision culling process. Hybrid collision culling that is a combination of bounding sphere and Oriented-Discrete Orientation Polytopes (Or-Dops) was implemented on multiple *.tri objects as outlined in previous section. Figure 11 shows hybrid collision culling implemented on a pair of colliding object – this is done to purposely highlight the concept. It was then tested against conventional bounding volume approach.



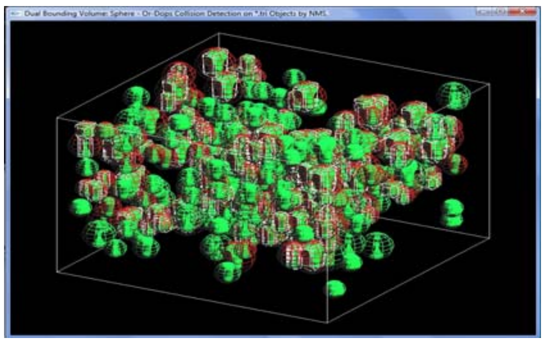


Figure 11. Hybrid Collision culling–note the Pairs with Sphere and or-Dops

Figure 12. Hybrid Collision Culling on 350 Objects

Initial result based on the frames per second (fps) of 500 objects undergoing rigid motion is shown as Figure 13. Simulation without bounding volume (thus no collision detection employed) was implemented as a control experiment. If hybrid collision culling is employed on a particular object of interest (labeled as '1ObjHybrid' in the graph), the frame rates can reach to nearly the performance of the control experiment, similar to the performance of sphere bounding volume. Fps performance dropped if all objects employed hybrid collision culling, but it still outperforms the homogeneous or-Dops implementation. Another approach was also tested where a two-pass test was employed (labeled as 'Seq Sp-OrD' in the graph). The first test to identify pairs that are likely to collide based on bounding sphere tests. These pairs are sent to the second pass where or-Dops tests will be conducted. It shows an improved performance if all objects needs to employ the hybrid collision culling method.

6. Conclusion

Results from the experiments shows that the hybrid collision culling method shows the potential of an option for a better collision culling technique for massive rigid body simulation compared to the homogeneous bounding volumes. However a detailed test like the total cost benchmarking test needs to be done to systematically evaluate this result.

Acknowledgment

This research is supported by Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education (MOHE), in collaboration with Research Management Centre (RMC), UTM. This paper is financially supported by Research University Grant (RUG) Program, Tier 2 (research grant 05J21).

References

[1] Q Avril, V Gouranton, B Arnaldi. Fast Collision Culling in Large-Scale Environments Using GPU Mapping Function. *Eurographics Symposium on Parallel Graphics and Visualization.* 2012.
[2] D Harmon, D Panozzo, O Sorkine, D Zorin. Interference-aware geometric modeling. *ACM Trans. Graph.* 2011; 30: 1-10.
[3] PM Hubbard. Collision Detection for Interactive Graphics Applications. *IEEE Transactions on Visualization and Computer Graphics*. 1995; 1: 218-230.
[4] NM Suaib, A Bade, D Mohamad. *Collision Detection: A Survey of Techniques and Application. Collision Detection for Real-Time Graphics: Series of Techniques.* Johor; UTM. 2009; 1: 1-18.
[5] NM Suaib, A Bade, AM Zin, TMT Sembok. *Oriented convex polyhedra for collision detection in 3D computer animation.* Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia Kuala Lumpur, Malaysia: ACM, 2006.
[6] J Klein, G Zachmann. *Time-critical collision detection using an average-case approach*. Proceedings of the ACM symposium on Virtual reality software and technology Osaka, Japan: ACM, 2003.
[7] C Mendoza, C O'Sullivan. *An Interruptible Algorithm for Collision Detection between Deformable Objects*. International Workshop on Virtual Reality and Physical Simulation (VRIPHYS'05). 2005: 73-80.
[8] T Giang, C O'Sullivan. *Closest Feature Maps for Time-Critical Collision Handling*. International Workshop on Virtual Reality and Physical Simulation (VRIPHYS'05). 2005: 65-72.
[9] T Giang, C O'Sullivan. Virtual Reality Interaction and Physical Simulation: Approximate collision response using closest feature maps. *Comput. Graph*. 2006; 30: 423-431.
[10] J Barbic, DL James. Subspace self-collision culling. ACM Trans. Graph. 2010; 29: 1-9.
[11] DS Coming, OG Staadt. Stride scheduling for time-critical collision detection VRST: ACM. 2007.
[12] DS Coming, OG Staadt. Stride Scheduling for Time-Critical Collision Detection. The Fourth International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'08). Atlanta, Georgia, 2008.
[13] GVD Bergen. Collision Detection in Interactive 3D Environments. *Elsevier.* 2004.