Quad Separation Algorithm for Bounding-Volume Hierarchies Construction in Virtual Environment Application

Hamzah Asyrani Sulaiman[1], Mohd Azlishah Othman[1], Mohd Muzafar Ismail[1], Mohamad Harris Misran[1], Maizatul Alice Binti Meor Said[1], Ridza Azri Ramlee[1], Abdullah Bade[2], Mohd Harun Abdullah[2]

[1]*Centre for Telecommunication Research and Innovation (CeTRI), Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia*

asyrani@utem.edu.my, azlishah@utem.edu.my, muzafar@utem.edu.my, harris@utem.edu.my, maizatul@utem.edu.my, ridza@utem.edu.my

[2] *School of Science and Technology, Universiti Malaysia Sabah, Beg Berkunci 2073, Kota Kinabalu, 88899 Sabah, Malaysia*

abade08@yahoo.com, harun@ums.edu.my

Abstract

In order to perform fast collision detection technique in Virtual Environment Application, researchers need to maintain the behaviour of the object itself before the objects come into contact. By enhancing the speed of intersection using Bounding-Volume Hierarchies technique, it helps to reduce the complexity and speed up the intersection process. Thus, in this paper we presented our novel algorithm for constructing Bounding-Volume Hierarchies using Quad Splitting method. Together with the Quad Splitting method is the implementation of Spatial Object Median Splitting technique (SOMS) in order to create a well-balanced tree for the object. We believed the key of performing fast intersection between two or more objects in Virtual Environment Application required a well-balanced and proper tree technique for Bounding-Volume hierarchies.

Keywords*: Collision Detection, Virtual Environment, Bounding-Volume Hierarchies*

1. Introduction

Simulating a virtual environment, consists of thousands of polygons, consumes a lot of computer resources[1, 2]. The rendering of thousands of polygons in a virtual environment requires the use of massive computer resources, *e.g.* memory and computer processing time. Depending on corresponding virtual environments, some applications such as three-dimensional (3D) games and simulation may require different computational systems, *e.g.* collision detection, culling system, lighting, shadowing, and pixel shader and anti aliasing, to increase the realistic of the application. By taking an example of commercialize 3D game application, the developer intended to develop a real-time adventure games with nice landscape. This action however is limited; as they could never produce a realistic adventure game, running in real-time, without sacrifice other computational systems.

In order to design a better system in visualizing the virtual environments, several important issues must be considered. For instance, most 3D applications need to maintain the suitable frame rates such as 30 – 60 frames per seconds (FPS) [2-6]. To achieve this requirement, the corresponding 3D games should be able to perform various optimization methods such as frustum culling [1, 7-10] and collision detection [5, 10-15] that could reduce the complexity of the virtual environments. However, it raised another issue where the balance between the virtual environment realism and the complexity of the applications must be considered [3, 10, 16].

The created dynamic virtual environments could be further improved by incorporating collision detection method for realistic effect. Dynamic virtual environments consists of several elements of object undergoing motion where few 3D objects such as car and train is moving including the collision detection system that has been inserted into simulation of the virtual environments in order to detect any potential of collision between these moving objects [48-50]. Meanwhile, collision detection has been widely discussed as one of the most required system to be incorporated into various 3D simulations. For example, the meteorite fall simulation might require sophisticated collision detection

method in order to determine the impact after-collision effect. Thus, the collision detection effects bring a virtual environment more toward a real world environment where user is able to navigate and feels the surrounding by using haptic device. It is known that detecting interference or collision between 3D models in virtual environments requires high computational cost [3, 17-20].

One of the common collision detection technique [21-27] widely discussed among researchers is Bounding-Volume Hierarchies (BVH). The hierarchical representation of BVH works by enveloping the 3D object with Bounding-Volume (BV). Then, starting from the root node (first BV), it divides the parent node into several parts. These parts are called leaf nodes. Hence, in order to use BVH for collision detection, both objects need to be bounded with BVH in order to perform the collision detection test. Collision detection only checks for intersection parts or nodes that is going to intersect with other nodes [6, 23, 28-30]. It greatly reduces computation cost instead of performing primitive-primitive testing between intersecting objects.

Performing BV-BV testing is faster compared to primitive-primitive testing as it checks for certain parts of BVH that is collided with another BVH bounded objects. Taking an example of two equal objects consisting of thousands of polygons, it can be bounded with only one simple BV. By determine the depth of BVH tree, the collision checking only needs to check for intersected parts thus reducing almost half of the triangles for each level of BVH tree. Therefore, the computation cost of detecting object interference for virtual environments can be lowered as the complexity of the object is reduced.

In this paper, an improved of BVH based collision detection technique is presented. Our contribution in this paper is to introduce a novel splitting technique of hierarchical representation to improve collision detection handling in virtual environment. An efficient splitting technique that increased the efficiency of BVH construction is presented. The section is organized as follows, section 2 discusses related work, section 3 discuss on the technique, and testing conditions. Two last sections will discuss the experimental results and conclusion of this research.

2. Research Study

The recent work of collision detection for rigid bodies is still an open research area instead of deformable bodies [30]. Some primary efforts of collision detection research are described in the next section. Collision detection for rigid bodies' simulation is one of the interesting research areas where it involves the location of the objects and not the objects itself. Rigid bodies stand for geometric models that are fixed and assumed to remain static except if there is some force is applied. In collision detection case, when two geometric models have collided, the system would notice that both objects cannot change it dimensions and sizes. Any deformation to rigid bodies is neglected because of this behaviour and the collisions only affect its location or the movement of both objects. This is because deformable models have deformation effects where their shape can be changed. Since earlier era of 3D simulation and animation, problems prevailed in detecting object interference parts where numerous attempts by researchers have been made to find the solution of the collision detection between rigid bodies. Baraff has made one of the earliest researches concerning detecting object interference between rigid bodies [31].

Later in 1993, M.C. Lin conducted a research of detecting object interference between rigid bodies. They defined that there are two types of contact between rigid bodies. It can be identified in M.C. Lin research as tangential collision and boundary collision [5]. Tangential collision happens when there is intersection between two surfaces at 90 degrees at geometric contact point. It means that the collision happens either from 90 degrees from above, bottom, right or left surfaces of corresponding polygons. Meanwhile boundary collision occurred when there is one object wants to check for potential collision when inside the object boundary. For example, a circle has it owns boundary made of radius given. Then if there is one point inside this radius has intersect with another circle, then the boundary collision has occurred. Figure 1 explains the situation.

Meanwhile, Redon in his research of collision detection explained that there were two way of performing collision detection between rigid bodies namely discrete collision detection and continuous collision detection [26]. According to his research, discrete collision detection (DCD) is performed by sampling the object motion toward the object that is going to be intersected and detect the object interpenetrations [6, 16, 17, 27, 30, 32-37]. DCD is the approach that been used by researchers to perform collision detection in term of speed. Meanwhile, continuous collision detection (CCD) computes from the first time of contact when object collided. It much slower compare to DCD methods

because CCD is more on accurate collision detection. The most common methods for rigid bodies are to use hierarchical representation which is BVH. Among the BV that has been used for constructing BVH are spheres [4, 32, 38-42], axis-aligned bounding boxes [27, 43, 44], oriented bounding boxes [45], and discrete orientation polytopes [46].

3. Bounding-Volume Hierarchies (BVH)

BVH is one of the hierarchical representation consists of the tree elements but in reverse order. At top of the BVH is a root node. From root node, it branches to create child nodes according to the type of the tree construction. The tree construction can be classified into several types. Among popular type of tree constructions that available are binary tree, quad-tree, oct-tree, and k-tree (where k is a number defined by the user or the system). These tree constructions may vary its advantages according to what kind of targeted application might require. Next, the tree branches until reaches certain level. The level of the tree construction can be defined either automatically by the system or application or manually by user. It stops according the stopping criteria set up by the system or the user. The last level of the tree is called leaf nodes. Leaf nodes consists of bounding-volume (BV) that has few set of triangles that is required for further collision testing involved primitive-primitive testing between set of triangles.

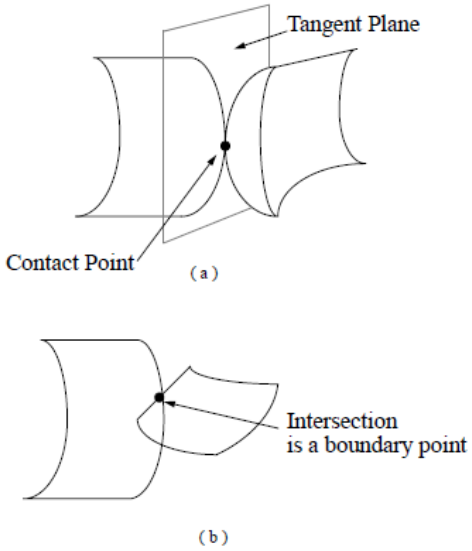

Figure 1. (a) Tangential Collision and (b) Boundary Collision [32]

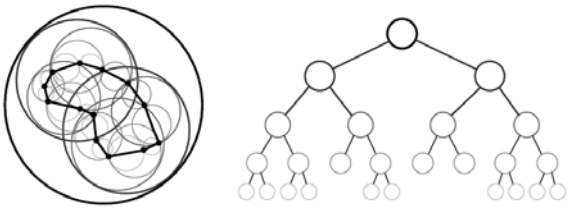

Figure 2. The left hand side image shows a BVH with Sphere BV while on the right hand side image, shows unbalanced hierarchical form using binary type hierarchy.

In order to perform collision detection test, both objects needs to be enclosed with BVH. When the collision occurs, the first intersection test is between root nodes. For example, when object A intersected with object B, the first test is between root node of object BVH A and object BVH B. If there is no collision occurred, the test will be stopped. However, if there is a

collision, the BVH traversal algorithm will be activated where it could traverse down the tree for any potential collision until it reaches leaf nodes. It is a recursive process where the exact collision needs to be determined. Both objects may not be collided if during the middle process of traversal algorithm checks for collision, it found that there is no collision occurs. Hence, the tree level must be measured carefully in order to increase the accuracy and speed of the collision detection method. It totally depends on how the 3D application required. Medical simulation might strive for very accurate collision detection method while computer games and animation may require very fast algorithm of collision detection method. BVH is just provided one of the effective ways to increase the detection process. Figure 2 depicts sphere BVH with unbalanced tree criteria.

4. Hierarchical Construction of Quad Axis Separation (QAS) based on Spatial Object Median Splitting (SOMS)

The important step in BVH tree construction is a splitting process. The splitting process usually is for top-down approach where a BV needs to be splitted into several parts depending on what type of tree construction that has been chosen. For binary type tree, the splitting process will divide the object BV into two separate parts. These parts consist of left and right nodes. Meanwhile, for other types of tree construction, the same process is used but only different in term numbers of nodes. There are several important steps for splitting process; Figure 3 illustrates the steps for BVH splitting process for tree construction using binary top-down approach.

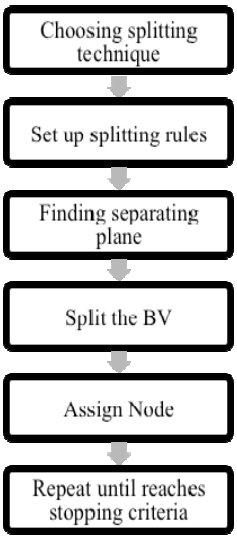

Figure 3. Splitting Procedure for BVH tree construction using binary type tree construction.

Our algorithm works by first implementing Axis-Aligned Bounding-Boxes (AABB) for corresponding object. By using min-max calculation, we enclosed each object with tight-fitted AABB and perform update if the object rotates in any direction. Once the object has been enclosed with an AABB, we can now proceed with our splitting algorithm called SOMS where it can properly divides object into two nodes called left and right node. SOMS exploit the capability of enclosing the midpoint of set of triangles with so-called “Inner AABB”. Inner AABB works by collecting the information of each triangle midpoint for corresponding node (all of them) and bound it with an AABB. By implementing this, we could get well-balanced tree compared to ordinary

Spatial Object Median Splitting method that using only the ordinary AABB to split the object into separate nodes. Figure 4 shows the corresponding Outer and Inner AABB for SOMS technique.

Compared to previous separation tree method, instead of separated the AABB into two separated nodes, we perform a modification of the splitting heuristic rules by approximating the corresponding set of midpoint triangle using optimize version of quad tree separation technique. Our optimized quad

tree separation technique can extend the capability of AABB BVH where it could provide the process of splitting process where it can generate four more AABB instead of two. This is a process called "Quad-Axis Separation" procedure.

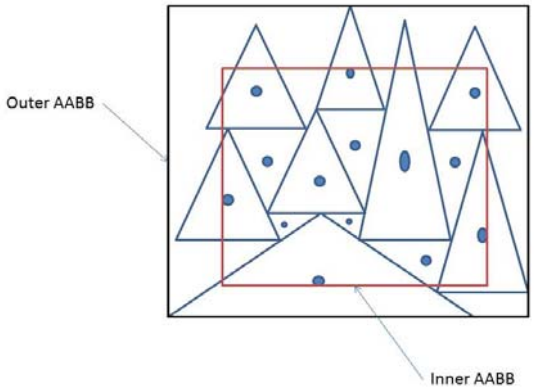

Figure 4. Inner and Outer AABB for SOMS technique **[**47]**.**

Beginning at the root nodes of tree

1. Enclose all midpoint with AABB
   a. Finding line equations based on min-max point
   b. Bundle each set of triangles into four group.
   c. Use SOMS for in the line midpoint to get better tree.
2. Done splitting
3. Enclose each set of triangles with AABB

Figure 5. Quad Axis Separation Technique

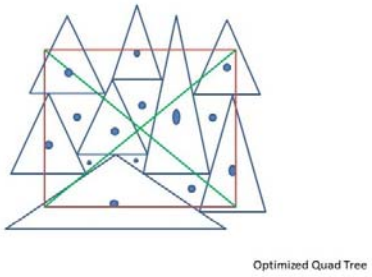

Figure 6. Quad-Axis Separation based on SOMS "Inner AABB"

5. Quad Axis Separation (QAS) Procedure

QAS works by setting up four nearest triangle midpoint to the each corner of vertices and then approximately divides into four AABBs. This process is to be repeated every even level until only one triangle left for each tree node. For any point that located exactly at the line equation, the point will be

divided using SOMS where QAS need to work closely with SOMS in order to achieve better trees. However, it depends heavily on the object triangle property before the object is to be split into four instead of two. Figure 5 shows an algorithm on how the process will be developed. Figure 6 and 7 shows an improvement over the previous technique.

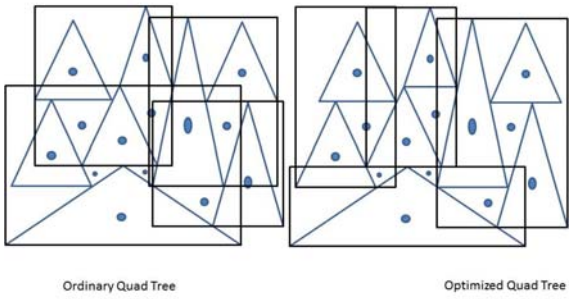

Figure 7. Quad-Axis Separation based on SOMS "Inner AABB"

Based on figure 5, the algorithms starts by getting a collection of midpoints for each triangle object. Once the data has been obtained and saved into temporary memory of the program, we continue to perform splitting using SOMS and manage the grouping of triangle into specific group. We will enclosed each node with AABB BVH once the splitting has been completed.

From the figure 6, we need to create an inner AABB in order to perform splitting method mentioned in figure 5 algorithm. By using this inner AABB information, we will perform Quad-Tree separation using SOMS and thus generated BVH tree with AABB just in figure 7. Figure 7 shows us how the differences between original quadtree BVH construction using normal splitting method and Optimized quadtree BVH using SOMS.

6. Real Case Implementation

Figure 8 shows that SOMS technique is faster than Spatial Median technique in term of construction time. SOMS is able to construct balanced BVH at fixed level 7 while Spatial Median become unbalanced when it reaches level 8. By analysing the Figure 8, we should notice that the BVH tree has been constructed multiple times (1000 times) and the average of these values is calculated.

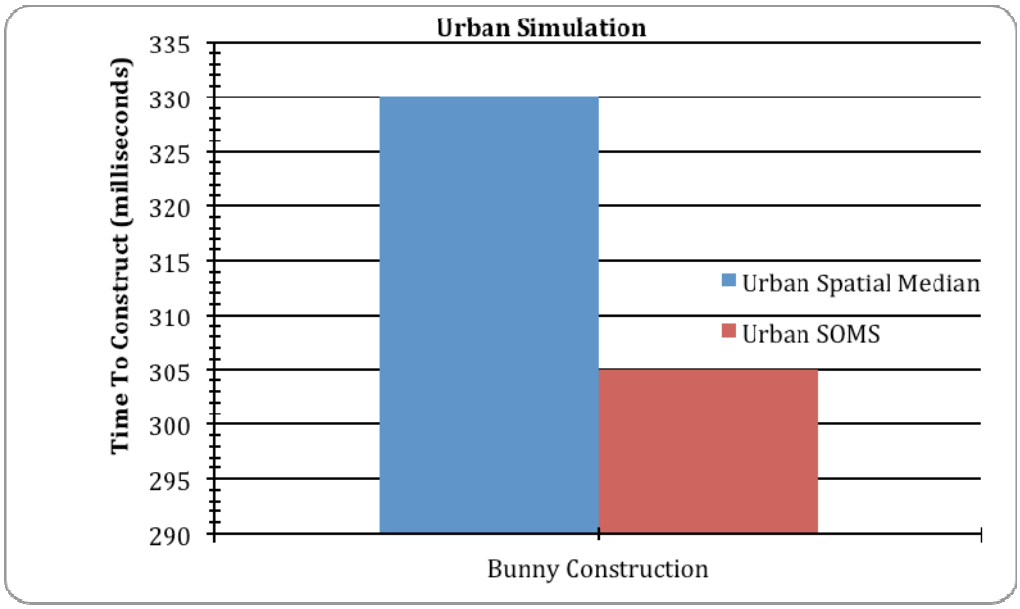

Figure 8. Average times to construct 1000 times BVH for 3DS Urban Simulation using Spatial Median and SOMS method. The Level of BVH is fixed at level 8 (Spatial Median Balanced BVH).

Hence, if the object consists hundreds of thousands polygon, we can concluded that the construction time for real application is dropped when using SOMS technique compare to Spatial Median technique. Apart from that, the Figure 8 explained the number of leaf nodes for both techniques are not in the same BV size.

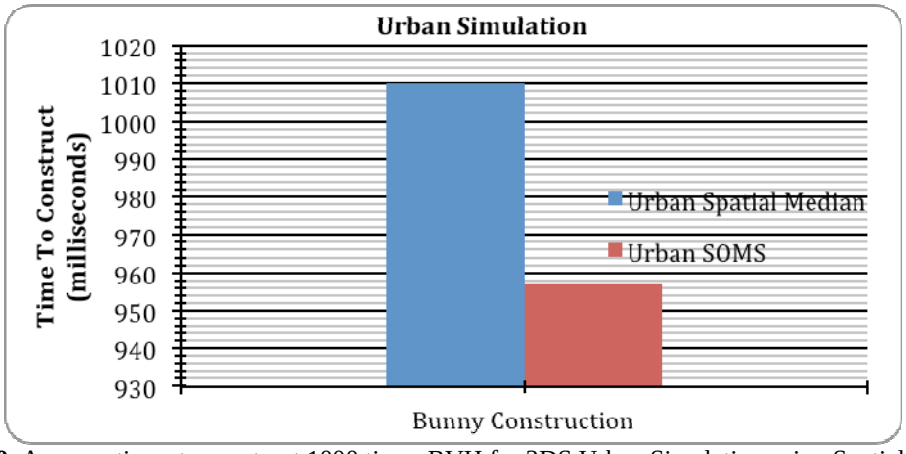

Figure 9. Average times to construct 1000 times BVH for 3DS Urban Simulation using Spatial Median and SOMS method. The Level of BVH is fixed at level 7 (SOMS Balanced BVH).

In this experiment of Figure 9, Spatial Median technique generated less nodes compared to SOMS technique. SOMS BVH tree needs to produce balanced BVH level tree thus level 7 BVH tree is supposed to generate $2^7 = 128$ nodes. Each node is supposed to fill with triangles. However for Spatial Median technique, BVH that is generated has fewer nodes than the actual total nodes (less than 128 nodes). Hence, this proof that even though the results showed that SOMS much faster than Spatial Median, SOMS still be able to construct faster BVH tree with 128 nodes compared to Spatial Median technique.

In previous statement, we can see how number of triangles for certain objects is really important for SOMS technique to press its advantage. From the previous figure below calculation of Total Time in Minute for BVH tree construction, SOMS technique proved that for 1000 times construction test, it

consumed 15 minutes and 57 seconds. Meanwhile for Spatial Median technique, the testing was running for 16 minutes and 50 seconds. The formula that we introduced here is to calculate the total time to construct 1000 times of BVH tree. It is our measurement to find BVH tree construction total time. From the testing that we run, we concluded that for any object that has hundreds of thousands polygon, SOMS be able to perform better than Spatial Median technique. By using below formula to calculate total time to construct:

$$Total\ Time \quad \frac{Total\ nodes \times 1000}{1000\ milliseconds \times 60}$$

Table 1. Construction Time

Technique	Time
Spatial Median	16 Minutes and 50 Seconds
Spatial Object Median (SOMS)	15 Minutes and 57 Seconds

From the calculation, it showed that SOMS technique produce a better BVH tree by achieving relatively less time compared to Spatial Median technique. As most of the complex environments consisting hundreds of thousands polygon, the number of time taken to construct BVH tree is going to increase. Hence, the outcome of this testing is to measure how efficient BVH tree construction if the complex environments consist a lot of triangles.

7. Conclusion

This technique intends to help reducing time to construct BVH while creating more balanced level or the tree reducing the use of heuristic determination. Since previous researchers used some heuristic to continue to split their tree, SOMS intends to reduce time to use heuristic for the node determination. This could benefit the real time construction when the real application of SOMS can be applied in deformable models instead of rigid bodies in future work.

8. Acknowledgment

We would like to thank to Universiti Teknikal Malaysia Melaka for give us support and GRAVSlab from Universiti Malaysia Sabah for research and moral support.

9. References

[1] P. Wonka, "Occlusion Culling for Real-Time Rendering of Urban Environments," PhD, Fakult¨at f¨ur Technische Naturwissenschaften und Informatik, Vienna Institute of Technology, Vienna, 2001.

[2] A. Brosnan, J. Hamill, and C. O'Sullivan, "Geometry Reduction for Urban Simulation on Handheld Devices," presented at the Sixth Eurographics Irish Chapter Workshop on Computer Graphics, 2005.

[3] S. Luiz Gonzaga da and M. Soraia Raupp, "Real-time generation of populated virtual cities," presented at the Proceedings of the ACM symposium on Virtual reality software and technology, Limassol, Cyprus, 2006.

[4] J. Spillmann, M. Becker, and M. Teschner, "Efficient updates of bounding sphere hierarchies for geometrically deformable models," J. Vis. Comun. Image Represent., vol. 18, pp. 101-108, 2007.

[5] H. A. Sulaiman, A. Bade, and N. M. Suaib, "Bounding-Volume Hierarchies Technique for Detecting Object Interference in Urban Environment Simulation," in Second International Conference on Environmental and Computer Science, 2009. ICECS '09, pp. 436-440, 2009.

[6] A. Bade, N. Suaib, M. Z. A, and T. S. T. M, "Oriented convex polyhedra for collision detection in 3D computer animation," presented at the Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, Kuala Lumpur, Malaysia, 2006.

[7] S. Yan, Z. Zhang, M. Peng, and M. Yang, "A Subjective Evaluation Method for Human-computer Interaction Interface Design Based on Grey Theory," in Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, pp. 220-220, 2006.
[8] J.-W. Chang, W. Wang, and M.-S. Kim, "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," Computer-Aided Design, vol. 42, pp. 50-57, 2010.
[9] L. Aiping and Z. Qinglin, "Correlations Among Cartilage Erosion, IgA Level, Red Blood Cell and Platelet Counts in 436 Rheumatoid Arthritis Patients with Path Analysis," in Bioinformatics and Biomedical Engineering , 2009. ICBBE 2009. 3rd International Conference on, pp. 1-3, 2009.
[10] T. Min, C. Sean, Y. Sung-Eui, and M. Dinesh, "Interactive continuous collision detection between deformable models using connectivity-based culling," presented at the Proceedings of the 2008 ACM symposium on Solid and physical modeling, Stony Brook, New York, 2008.
[11] A. Garcia-Alonso, Nicol\, \#225, s. Serrano, and J. Flaquer, "Solving the Collision Detection Problem," IEEE Comput. Graph. Appl., vol. 14, pp. 36-43, 1994.
[12] D. O. Tuft, "A System For Collision Detection Between Deformable Models Built On Axis Aligned Bounding Boxes And Gpu Based Culling," Master of Science, Department of Computer Science, Brigham Young University, Brigham, 2007.
[13] V. R. Kamat and J. C. Martinez, "Interactive collision detection in three-dimensional visualizations of simulated construction operations," Engineering with Computers, vol. 23, pp. 79-91, 2007.
[14] S. Trenkel, R. Weller, and G. Zachmann, "A Benchmarking Suite for Static Collision Detection Algorithms," presented at the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Plzen, Czech Republic, 2007.
[15] G. Jianhong, H. Hanwu, Z. Wenxuan, and L. Yanfei, "Research on real-time collision detection for vehicle driving in the virtual environment," in International Conference on Information and Automation, 2008. ICIA 2008. , pp. 1834-1839, 2008.
[16] Y. XianYi and C. Guo, "Human-Computer Interaction Design in Product Design," in Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on, pp. 437-439, 2009.
[17] W. Ping, L. Jixiang, Y. Long, and L. Changde, "Research on the virtual 3D Human -computer Interaction based on the 3G terminal cross-platform," in Computer-Aided Industrial Design & Conceptual Design, 2009. CAID & CD 2009. IEEE 10th International Conference on, pp. 481-484, 2009.
[18] C. Gong, "Human-Computer Interaction: Process and Principles of Human-Computer Interface Design," in Computer and Automation Engineering, 2009. ICCAE '09. International Conference on, pp. 230-233, 2009.
[19] Z. Kunxia, Z. Meiling, G. Zhengqin, and W. Zuolong, "Realization of Human-Computer Interaction Functions in Virtual Reality," in Research Challenges in Computer Science, 2009. ICRCCS '09. International Conference on, pp. 229-231, 2009.
[20] R. G. Laycock, G. D. G. Ryder, and A. M. Day, "Technical Section: Automatic generation, texturing and population of a reflective real-time urban environment," Comput. Graph., vol. 31, pp. 625-635, 2007.
[21] K. Somchaipeng, K. Erleben, and J. Sporring, "A multi-scale singularity bounding volume hierarchy," in Proceedings of WSCG, pp. 179-186, 2005.
[22] K. Erleben, J. Sporring, K. Henriksen, and K. Dohlman, Physics-based Animation (Graphics Series): Charles River Media, Inc., 2005.
[23] A. Nguyen, "IMPLICIT BOUNDING VOLUMES AND BOUNDING VOLUME HIERARCHIES," Doctor of Philosophy, Stanford University, 2006.
[24] J.-W. Chang, W. Wang, and M.-S. Kim, "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," Computer-Aided Design, vol. In Press, Corrected Proof, 2009.
[25] Z. Kun, H. Qiming, W. Rui, and G. Baining, "Real-time KD-tree construction on graphics hardware," ACM Trans. Graph., vol. 27, pp. 1-11, 2008.
[26] N. M. Suaib, A. Bade, D. Mohamad, and H. A. Sulaiman, "On Faster Bounding Volume Hierarchy Construction for Avatar Collision Detection," in International Conference on Computer Technology and Development, 2009. ICCTD '09, pp. 430-434, 2009.

[27] C. Tu and L. Yu, "Research on Collision Detection Algorithm Based on AABB-OBB Bounding Volume," in First International Workshop on Education Technology and Computer Science, 2009. ETCS '09. , pp. 331-333, 2009.
[28] G. Zachmann, "Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques," Department of Computer Science, Darmstadt University of Technology, Germany, 2000.
[29] N. M. Suaib, A. Bade, and D. Mohamad, "Collision Detection Using Bounding-Volume for avatars in Virtual Environment applications," in The 4th International Conference on Information & Communication Technology and Systems, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, pp. 486 – 491, 2008.
[30] S. H. Kockara, T.; Iqbal, K.; Bayrak, C.; Rowe, Richard;, "Collision Detection - A Survey," presented at the IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC., 2007.
[31] D. Baraff, "Analytical methods for dynamic simulation of non-penetrating rigid bodies," presented at the Proceedings of the 16th annual conference on Computer graphics and interactive techniques, 1989.
[32] R. d. S. Rocha and R. Maria Andre'ia Formico, "An evaluation of a collision handling system using sphere-trees for plausible rigid body animation," presented at the Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, 2008.
[33] J. T. Klosowski, M. Held, J. S. B. Mitchell, H. Sowizral, and K. Zikan, "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," IEEE Transactions on Visualization and Computer Graphics, vol. 4, pp. 21-36, 1998.
[34] S. Redon, A. Kheddar, and S. Coquillart, "CONTACT: arbitrary in-between motions for collision detection," in 10th IEEE International Workshop on Robot and Human Interactive Communication, pp. 106-111, 2001.
[35] S. Redon, A. Kheddar, and S. Coquillart, "Fast Continuous Collision Detection between Rigid Bodies," Computer Graphics Forum, vol. 21, pp. 279-287, 2002.
[36] E. G. Gilbert and C. P. Foo, "Computing the distance between general convex objects in three-dimensional space," Robotics and Automation, IEEE Transactions on, vol. 6, pp. 53-61, 1990.
[37] S. Lokesh, G. Balakrishnan, S. Malathy, and K. Murugan, "Computer Interaction to human through photorealistic facial model for inter-process communication," in Computing Communication and Networking Technologies (ICCCNT), 2010 International Conference on, pp. 1-7, 2010.
[38] G. Bradshaw and C. O'Sullivan, "Sphere-tree construction using dynamic medial axis approximation," presented at the Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, San Antonio, Texas, 2002.
[39] T. Larsson, T. Akenine-Möller, and E. Lengyel, "On Faster Sphere-Box Overlap Testing," Journal of Graphics Tools, vol. 12, pp. 3-8, 2007.
[40] R. Weller and G. Zachmann, "Inner Sphere Trees," Clausthal University of Technology, Clausthal-Zellerfeld, Germany 2009.
[41] A. Benitez, M. d. C. Ramirez, and D. Vallejo, "Collision Detection Using Sphere-Tree Construction," presented at the Proceedings of the 15th International Conference on Electronics, Communications and Computers, 2005.
[42] B. Gareth and O. S. Carol, "Adaptive medial-axis approximation for sphere-tree construction," ACM Trans. Graph., vol. 23, pp. 1-26, 2004.
[43] R. e. Weller, J. Klein, and G. Zachmann, "A Model for the Expected Running Time of Collision Detection using AABB Trees," in Eurographics Symposium on Virtual Environments (EGVE), Lisbon, Portugal, 2006.
[44] X. Zhang and Y. J. Kim, "Interactive Collision Detection for Deformable Models Using Streaming AABBs," IEEE Transactions on Visualization and Computer Graphics, vol. 13, pp. 318-329, 2007.
[45] S. A. Gottschalk, "Collision queries using oriented bounding boxes," The University of North Carolina at Chapel Hill, 2000.
[46] M. d. Berg, H. Haverkort, and M. Streppel, "Efficient c-Oriented Range Searching with DOP-Trees " vol. Volume 3669/2005, pp. 508-519, 2005.

[47] H. A. Sulaiman, A. Bade, and N. M. Suaib, "Fast Traversal Algorithm for Detecting Object Interference Using Hierarchical Representation between Rigid Bodies," presented at the Proceedings of the 2010 Second International Conference on Computer Research and Development, 2010.
[48] Yumei Xiong, Chao Li, "A new Parallel Collision Detection Algorithm Based on Genetic Algorithm", JCIT: Journal of Convergence Information Technology, Vol. 8, No. 3, pp. 260 ~ 268, 2013.
[49] Xinlu Ma, Yeren Liu, Jiang Cheng, "Analytic and Heuristic Methods of Collision Detection for Curve-Moving Vehicle Based on Cooperative Collision Warning System", JCIT, Vol. 8, No. 2, pp. 244 ~ 253, 2013.
[50] Yumei Xiong, Yimin Chen, "An Improved Bounding Box Algorithm to Collision Detection for Virtual Environment ", JDCTA, Vol. 5, No. 2, pp. 230 ~ 237, 2011.