Quad Axis Separation Framework for Bounding-Volume Hierarchies Construction

Hamzah Asyrani Sulaiman, Abdullah Bade, Mohd Harun Abdullah

Abstract—The construction of Bounding-Volume Hierarchies (BVH) for Virtual Environment application has been varied from the rigid bodies application type to the deformable bodies application. Numerous technique and specific instruction has been given from several researchers in order to make sure that the BVH can suite their application without any restriction. In this paper, we explore the capability of BVH using a technique called Quad Axis Separation Technique (QAS) that could efficiently create full-blown hierarchical tree using approximation of separating axes theorem for Virtual Environment. A theoretical implementation is carried out with standard experimental that is also been used by researcher to test their BVH in the Virtual Environment. We also believed that QAS could provide fast and efficient hierarchical tree construction and also enhance the speed and accuracy of the collision detection technique

Index Terms— Bounding-Volume Hierarchies, Collision Detection, Bounding-Volume, Virtual Environment

1 Introduction

In developing 3-Dimensional (3D) world, various considerations must be taken into account before the simulation is going to be online for real-time visualization. Number of polygons for each Virtual Environment application depends heavily on the technique that the designer is going to develop and propose. One might want to build a virtual city just to highlight the environment for virtual tourism purpose while others is really pun into concentration on building a real-time simulation consisting complex objects that could test their propose technique. For collision detection technique, the researcher can just test the contact between two polyhedral, which is a rigid bodies simulation. Hence, it is essential for researchers or designers to properly getting the information require when building Virtual Environment (VE) application such as medical simulation and computer games development [42-43].

Complexity of the VE can be varied according to the authors or the designers. Some might saying that 10000 triangles are complex enough for VE application while there are many simulation exists that represented more than millions of triangles. It seems that the number of polygons does not necessarily represent the complex environments but the detail of the VE itself is the important point to consider as complex environments.

The smallest thing in VE is consists of set of point where this set of points will be connected and created a triangle. Set of triangle then is also connected where it creates an object. Thus, performing collision detection where two or more object overlapped into each other is a huge task. Imagine if each object contains a lot of triangles and need to test every single triangle for overlap. Researchers have introduced a simple and fast enclosed area for each object called Bounding-Volume (BV). This BV is then is optimized by the researchers by producing Bounding-Volume Hierarchies (BVH) for more accurate collision detection testing.

Bounding-Volume Hierarchies (BVH) stand for hierarchical solution for VE where each single object in VE will be bound with Bounding-Volume (BV). Then, each BV will be split into several smaller parts according to the type of trees such as binary, quad, or k-tree splitting. The cost of splitting the BV is depending on how well the separating axes theorem that has been used in the BVH construction. Sometimes the construction can lead into infinite level of splitting process and thus the researchers must limit the level of hierarchical tree by using heuristic determination. Later, this BVH that enclosed the object will be tested in VE where the collision detection will be activated.

Hence in this research paper, we would like to propose a method that could optimize the Bounding-Volume Hierarchies construction in both pre-processing and post-processing time. By optimizing the way on how separating axes theorem works, we could enhance the capability of BVH construction in term of speed and efficiency when performing collision detection in VE. Section II will describe related works about the BVH and collision detection method while Section III describes about the property of BVH and BV. Later in Section IV, we suggested an algorithm for optimizing separation axes theorem for better BVH tree construction. Section V consisted of experiments procedure for tree construction by comparing with common separation axes theorem techniques. In Section VI we conclude and give discussion about future work.

2 Related Word

2.1 Discrete and Continuous Collision Detection

In VE, common solution that has been used for collision detection technique is grid-based partitioning, spatial partitioning and Bounding-Volume Hierarchies. However, grid-based solution is almost the same as spatial partitioning where grid-based solution can be used either for single object or whole VE. Meanwhile Bounding-Volume

Hierarchies has been studied in various fields of computer graphics such as ray-tracing, mesh subdivision and collision detection.

Research on collision detection between rigid body's simulation and deformable body's simulation has been studied for almost few decades ago. As the VE itself becoming far better than previous decade where most film has embedded 3D simulation, we found that the research of collision detection has become more unique with several of techniques proposed by researchers. In VE, a rigid body is defined as a model that cannot be modified even some force has been applied. But under certain circumstance, it can be destructible where the behavior of rigid body has turned into deformable bodies which can be deformed or modified. Thus, it is simple to conduct an experiment using rigid body's simulation but the hardest part is to still to find the accurate or point of contact for the object. Rigid bodies collision detection materials can be found by in many literatures now days [2-6].

Later in 1993, M.C. Lin conducted a research of detecting object interference between two rigid bodies. They defined that there are two types of contact between rigid bodies that could be identified in M.C. Lin research as tangential collision and boundary collision [7]. 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.

Whilst, Redon explained that there are two common types of performing collision detection between rigid bodies namely discrete collision detection and continuous collision detection [8]. 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 [5, 9-18]. In computer games, DCD is the most popular solution where it can perform fast and accurate enough between objects collision.

CCD methods primarily improve the accuracy of collision detection compared to DCD methods but it also genuinely took longer computation time. Therefore, in computer or console game applications, CCD technology has not been used actively in such area. However, there are many approaches that has been used to accelerate the performance of CCD by creating a specific instruction or algorithm for specific applications on certain types of models (e.g., rigid objects, articulated bodies, meshes with fixed topology), utilizing the computation power of GPUs [4, 19-22], and introducing efficient culling methods [23]. But again, those algorithms suffer from computation cost that may take hundreds of milliseconds and, even, a few seconds on performing CCD for deforming models for example that may consists of hundreds of thousands triangles.

2.2 Bounding-Volume Hierarchies

BVH allow the intersection occurs without searching for non-colliding pairs from the hierarchy tree. For example, given two objects with their BVH, when root of the hierarchies do not intersect, the calculation will not be done for the both objects. However, when the root of both hierarchies intersects, it checks for intersection between one root of the hierarchies' tree and the other children of objects hierarchies' tree.

Various works has been done in sphere, AABB, and OBB tree construction where the cost of constructing these trees is practically low computation cost compared to other type of BVH trees [5, 9, 24-29]. Since the type of the tree also play important role in determine the speed and the accuracy when performing collision detection algorithm. Some optimization for them has been applied with specific instruction.

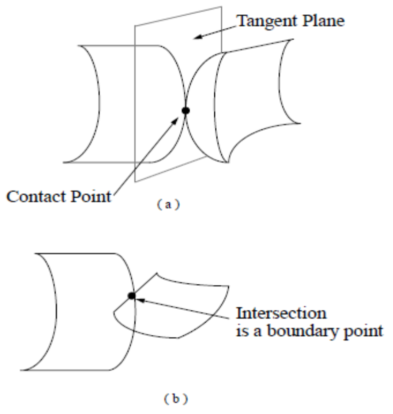

Fig. 1 (a) Tangential Collision and (b) Boundary Collision [7]

3 Bounding Volume Hierarchies for Collision Detection in Virtual Environments

Hierarchical tree consists of several nodes where each node can be labeled as root, child node, and leaf node. Depending on the type of the tree, it can be divided into several child nodes. Each child node will then act as subroot where it also can be divided into several child nodes. Once the tree has been properly managed where it stops to divide according to the user specification, each node will be having at least several leaf nodes or at least one leaf node. Figure 1 shows the BVH construction level.

By visualizing it into Virtual Environment (VE), we can have one Bounding-Volume (BV) to enclose each object. It might Sphere BV [28, 30-32], AABB[9, 29, 33], OBB [9, 34-36], K-Dop[12, 24], ellipsoid[37] OR-Dop[11] or other types of BV. The usage of BV depends heavily on the target application that we want to develop. Fast response collision detection in computer games development want to have fast collision detection method and thus Sphere BV, AABB, or OBB can be considered as an ideal BV whereas medical simulation or deformable bodies simulation might requires K-Dop and ellipsoid for better accuracy and moderate speed of detection. There were certain cases where AABB and OBB also can produce accurate collision detection method.

Meanwhile, the type of tree for the BVH construction must be determined at first place. For binary construction, each root will be divided into left and right nodes and the process continue until the stopping criteria has stopped the process from infinitely splitting. As each object contains a lot of triangle and most of them will never has the same triangle size. The splitting process might not be able to stop when the user wants it due to the bad heuristic usage. For example, we have conducted an experiment that shows spatial median splitting for AABB cannot properly split into balanced hierarchical tree until certain level where the splitting only lead to the infinite loop of splitting process [6].

The information stored in each tree node usually contains the data of vertices and indices or set of triangles. This information will be used to create another BV that only enclosed the set of triangles. This process will be continuing until the reasonable level has been achieved based on the heuristic determination that has been setup by the program. It is essential to provide good heuristic determination to develop proper and well balanced tree. Balanced tree is required for faster collision detection checking by reducing number of nodes that needs to be checking. We have performed a simple testing with spatial object median splitting method (SOMS) where it developed well balanced tree compared to common spatial median splitting method [7].

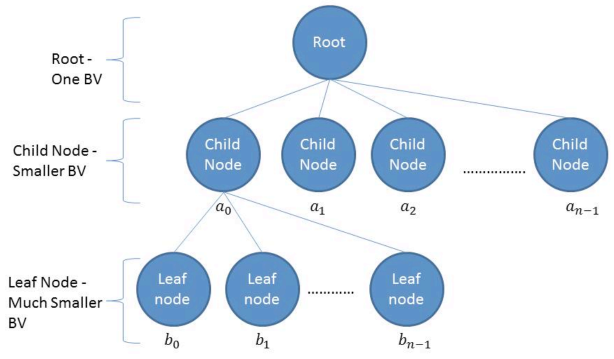

Fig. 2 BVH Tree Construction Level

3.1 BVH Trees

There are multiple types of BVH construction that can be used to represent hierarchical representation. Among those popular tree construction is binary tree. Binary tree has some nodes definition:-

a) Root Nodes – The topmost node in binary tree is called root node. Root node actually does not have any parents. It is the starting node where all the operations of tree will begin normally. Root Node is connected to each child nodes downward of root node by using branch or subtree connector. Branch of subtree connecter is one of the important elements that connecting each nodes with their parent nodes.
b) Leaf Nodes – The bottommost node in binary tree is call leaf nodes. It does not have any child nodes and mostly contain the last value or conditions. In BVH, it may contain few triangles or maybe one triangle.
c) Internal Nodes – The functional node where it has both parent nodes and child nodes. It is not leaf nodes as it still has child nodes and linked by parent node.
d) Subtree – A portion of a tree data structure that can be viewed as a complete tree in itself. It represents some other parts of the tree that can be manipulated by user in programming.

There are few common operations involved in construction and implementation of binary tree for BVH construction. Among them are enumerating all the items, searching for an item which is bounding-volume, removing a whole section of a tree for non-intersected area when performing collision detection (called pruning), traverse down from the root to leaf nodes or leaf nodes to the root, and report any intersected area between two intersected BVH that possibility in collision.

3.2 Splitting Rules for BVH Construction

The first step of BVH construction is choosing the splitting rules that will be divided the root node into two equal parts using binary type tree. Then, by using longest axis separating plane technique, the object will be divided at the longest axis at the splitting point based on the splitting rules. According to [38, 39], there was a three common solution of choosing the splitting rules for BVH construction:

a) Median of the object coordinates (object median) – split at the object median (evenly distributed parts) resulting a well-balanced tree.

b) Mean of object coordinates (object mean) – split at the object mean

c) Median of the bounding-volume projection extents (spatial median) – splitting the volume into two equal parts

3.3 BVH Intersection and Traversal

BVH allow the intersection occurs without searching for non-colliding pairs from the hierarchy tree. For example, given two objects with their BVH, when root of the hierarchies do not intersect, the calculation will not be done for the both objects. However, when the root of both hierarchies intersects, it checks for intersection between one root of the hierarchies' tree and the other children of objects hierarchies' tree. In this case, it recursively checks again whether there is intersection between both objects at middle level until if found the correct intersection. Figure 2 depicts the basic algorithms for detecting collision between two hierarchies [1, 40].

```
Beginning at the root nodes of two given trees
Check for intersection between two parent nodes
        If there is no intersection between two parents
Then stop and report "no collision"
        Else check all children of one node against all Children of the other node
        If there is intersection between any children
                Then If at leaf nodes
        Then report "possible collision"
        Else go to Step 4
Else skip and report "no collision"
```

Fig. 3 BVH traversal algorithm proposed by [1] for collision detection between two BVH

4 QUAD AXIS SEPARATION TECHNIQUE

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 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 process called "Quad-Axis Separation" procedure.

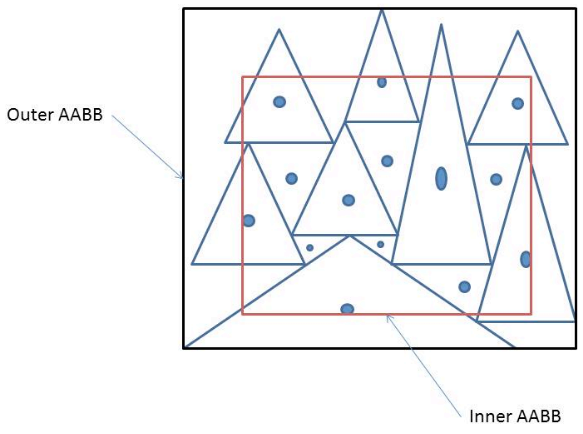

Fig.4 Inner and Outer AABB for SOMS technique [41].

4.1 Quad Axis Separation 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. 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

```
Beginning at the root nodes of tree
1.Create AABB
    a. Enclose all midpoint with AABB
    b. Create Axis splitting
        Finding Four Separating Line
        based on min-max point
    c. Bundle each set of triangles into
       four group
    d. Done splitting
2. Enclose each set of triangles with AABB
3. Repeat process
```

Fig. 5 Quad Axis Separation Technique

Figure 6, 7 and 8 shows an improvement over the previous technique.

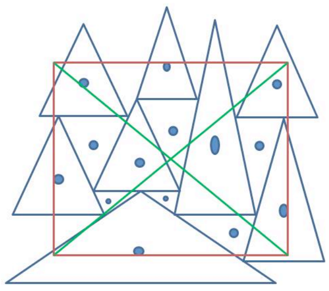

Fig. 6 Quad-Axis Separation based on SOMS "Inner AABB"

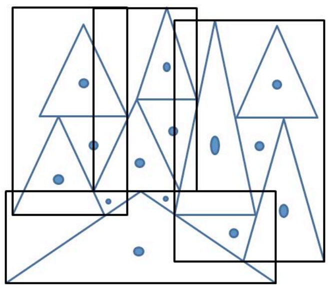

Fig. 7 AABB based on our optimized Quad Tree

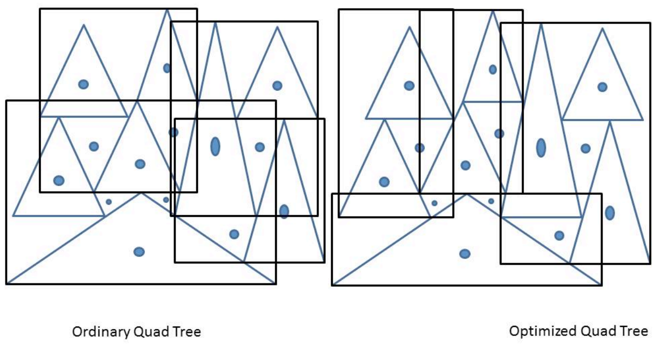

Fig. 8 Comparison with the ordinary Quad Tree. Better enclosing AABB with our optimization

5 Discussion and Analysis

Given the analysis from the figure, we can perform theoretical test for the corresponding algorithm when performing collision detection test compared to previous quad tree construction technique. Table 1 depicts the comparison test of triangle-triangle intersection test to consider. From the table, it depicts that ordinary generate more set of triangles compared to the QAS Quad Tree. It is important to reduce the number of test for triangle-triangle intersection test for collision detection.

TABLE I. Triangle-Triangle Intersection Test to Consider (Theoritical)

Technique			Tree			
	Total Triangle		*1*	*2*	*3*	*4*
Quad Tree	14		5	8	3	5
QAS Quad Tree	14		3	4	3	5

Only Tree no.4 consists of the same triangle in AABB tree

6 Conclusion And Future Work

We presented a theoretical framework of QAS implementation for ordinary Quad-Tree construction. In the next phase of our research, we are going to perform real-time visualization test using number of complex polygons object in 3D virtual environment application. We will also combine with our improve version of collision detection distance computation algorithm that will work together with this technique. The limitation however is concentrated on the triangle itself as the object might defer with one another. The size of triangle play important roles in determines the success rate of this technique.

Acknowledgment

This project is being funded by Universiti Teknikal Malaysia Melaka Short Term Grant PJP/2012/(2D)/S01003 would like to extend our gratitude to members of gravslab.com for research support and joint research university between Universiti Teknikal Malaysia Melaka, Universiti Malaysia Sabah, Universiti Teknologi Malaysia and Universiti Tun Abdul Razak (UNITAR).

References

[1] A. Nguyen, "IMPLICIT BOUNDING VOLUMES AND BOUNDING VOLUME HIERARCHIES," Doctor of Philosophy, Stanford University, 2006.

[2] 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.

[3] K. H. James, "Realistic animation of rigid bodies," presented at the Proceedings of the 15th annual conference on Computer graphics and interactive techniques, 1988.

[4] S. Rahul and L. Adam, "Rigid body collision detection on the GPU," presented at the ACM SIGGRAPH 2006 Research posters, Boston, Massachusetts, 2006.

[5] 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.

[6] H. A. Sulaiman, A. Bade, and N. M. Suaib, "Balanced hierarchical construction in collision detection for rigid bodies," in *Science and Social Research (CSSR), 2010 International Conference on*, 2010, pp. 1132-1136.

[7] 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*, 2009, pp. 436-440.

[8] 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*, 2009, pp. 430-434.

[9] 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.* , 2009, pp. 331-333.

[10] 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.

[11] 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.

[12] 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.

[13] 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, 2001*, 2001, pp. 106-111.

[14] 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*, 2009, pp. 437-439.

[15] 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*, 2009, pp. 481-484.

[16] S. Redon, A. Kheddar, and S. Coquillart, "Fast Continuous Collision Detection between Rigid Bodies," *Computer Graphics Forum,* vol. 21, pp. 279-287, 2002.

[17] 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.

[18] 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*, 2010, pp. 1-7.

[19] 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.

[20] P. Kipfer, "LCP Algorithms for Collision Detection Using CUDA," in *GPU Gems 3*. vol. 3, H. Nguyen, Ed., ed, 2007.

[21] Y. Han and P. Shi, "An improved ant colony algorithm for fuzzy clustering in image segmentation," *Neurocomputing,* vol. 70, pp. 665-671, 2007.

[22] L. Hanwen, W. Yi, and M. Guanghui, "A CPU-GPU hybrid computing framework for real-time clothing animation," in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, 2011, pp. 391-396.

[23] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "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.

[24] M. d. Berg, H. Haverkort, and M. Streppel, "Efficient c-Oriented Range Searching with DOP-Trees " vol. Volume 3669/2005, pp. 508-519, 2005.

[25] J. Bittner, P. Wonka, and M. Wimmer, "Visibility preprocessing for urban scenes using line space subdivision," in *Pacific Graphics 2001 (Ninth Pacific Conference on Computer Graphics and Applications*, 2001, pp. 276-284.

[26] M. Ernst and G. Greiner, "Early Split Clipping for Bounding Volume Hierarchies," in *Interactive Ray Tracing, 2007. RT '07. IEEE Symposium on*, 2007, pp. 73-78.

[27] M. A. Otaduy, O. Chassot, D. Steinemann, and M. Gross, "Balanced Hierarchies for Collision Detection between Fracturing Objects," in *Virtual Reality Conference, 2007. VR '07. IEEE*, 2007, pp. 83-90.

[28] R. Weller and G. Zachmann, "Inner Sphere Trees," Clausthal University of Technology, Clausthal-Zellerfeld, Germany2009.

[29] 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.

[30] 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.

[31] 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.

[32] T. Larsson, T. Akenine-Möller, and E. Lengyel, "On Faster Sphere-Box Overlap Testing," *Journal of Graphics Tools,* vol. 12, pp. 3-8, 2007.

[33] 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.

[34] 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.

[35] 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.

[36] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: a hierarchical structure for rapid interference detection," presented at the Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996.

[37] L. Shengjun, C. L. W. Charlie, H. Kin-Chuen, J. Xiaogang, and Z. Hanli, "Ellipsoid-tree construction for solid objects," presented at the Proceedings of the 2007 ACM symposium on Solid and physical modeling, Beijing, China, 2007.

[38] G. V. D. Bergen, *Collision Detection in Interactive 3D Environments*. United States of America: Elsevier, Inc., 2004.

[39] C. Ericson, *Real-Time Collision Detection (The Morgan Kaufmann Series in Interactive 3-D Technology) (The Morgan Kaufmann Series in Interactive 3D Technology)*: Morgan Kaufmann Publishers Inc., 2004.

[40] 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.

[41] 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.

[42] A. M. Adeshina, R. Hashim, N. E. A. Khalid, and S. Z. Z. Abidin, "Hardware-Accelerated Raycasting: Towards an Effective Brain MRI Visualization," Journal of Computing, vol. 3, p. 7, 2011.

[43] S. Goswami and A. Chakrabarti, "Quartile Clustering: A quartile based technique for Generating Meaningful Clusters," Journal of Computing, vol. 4, p. 8, 2012.

Hamzah Asyrani Sulaiman is a member of the IEEE (starting 2010) and the IEEE Computer Society(starting 2012). He is also a researcher and lecturer at Universiti Teknikal Malaysia Melaka. His specialization is Computer Graphics and Visualization. He has completed his Master thesis on Computer Graphics in 2010 and Bachelor of Engineering (Computer) in 2007. Current Research Group is GRAVSLAB (www.gravslab.com)

Dr. Abdullah Bade holds MSc degree by research in Computer Science from Universiti Teknologi Malaysia in 2003. He obtained his Doctorate degree in Industrial Computing (Computer Graphics and Visualization) from Universiti Kebangsaan Malaysia in 2008. He has been actively involved in research and manages to secure several research grants from Ministry of Higher Education and Ministry of Science and Technology Malaysia. His research interest lies particularly in developing optimized algorithm for collision detection between objects, Deformable Body Simulation, Serious Games Simulation, Cloth Simulation and Crowd Simulation. Currently, he is part of the School of Science and Technology, UMS and appointed as a Senior Lecturer and Head of Programmes for Mathematics with Computer Graphics. He spends most of his leisure time on listening soft music, surfing internet, reading and travel. Current Research Group is GRAVSLAB (www.gravslab.com)

Prof. Dr. Mohd Harun Abdullah is currently a Deputy Vice Chancellor of Universiti Malaysia Sabah and from School of Science and Technology, UMS