Distance Approximation using Pivot Point in Narrow Phase Collision Detection

Hamzah Asyrani Sulaiman[1], Mohd Azlishah Othman[2], Lizawati Salahuddin[3], Muhammad Noorazlan Shah Zainudin[4], Sani Irwan Md Salim[5], Mohd Muzafar Ismail[6]
Universiti Teknikal Malaysia Melaka
Durian Tunggal, Melaka, Malaysia
[1]asyrani@computer.org, [2]azlishah@utem.edu.my, [3]lizawati@utem.edu.my, [4]noorazlan@utem.edu.my, [5]sani@utem.edu.my, [6]muzafar@utem.edu.my,

Abdullah Bade[7], Mohd Harun Abdullah[8]
School of Science and Technology
Universiti Malaysia Sabah
Kota Kinabalu, Sabah
[7]abade08@yahoo.com, [8]harun@ums.edu.my

***Abstract*—Discrete and Continuous Collision Detection is two common fields in Collision Detection research area where it helps to determine time and point of contact when two object intersect. Each technique increase speed and accuracy of the simulation itself but depending on application, we need to have specific solution of collision detection technique. Most computer games and simulation maintain speed as the main important elements while others such as medical and mechanical simulation needs to have a very high precision collision detection technique. Thus, an algorithm for the optimal distance computation algorithm for continuous collision detection is shown in this paper. The basic idea is to use an AABB for both object triangles and creating a moveable origin point called Dynamic Origin Point (DyOP). DyOP created by using minimum and maximum point of both AABBs where it dynamically changes whenever the object move. This is a novel algorithm that works better than the previously known Gilbert Keerthi-Johnson algorithm and Lin-Canny algorithm where it helps to reduce the complicated test and implementation. We have shown that our technique is performed faster than the previous algorithms by increasing speed and nearly approximate the good distance between two nearly intersected triangles.**

Keywords— collision detection, virtual environment, distance computation, computer graphics

I. 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.

Determining the precise contact between two or more polyhedral is not an easy task as every time steps and movement of the objects must be calculated and computed. Hence, the collision detection system must be properly installed in the simulation or computer games in order to maintain the realism of the surrounding environments. However, the problem exists where the collision response can only manage to react once the collision has been collided before sending information that the objects has come into contact. The matter of this event is how long computer can receive input from the collided objects back to the collision detection mechanism. Important input such as time of contact, location of the contact and time lapse between reporting this event back is required in order to maintain the stability of the simulation. [1-7].

In this research, we have proposed a theoretical framework on how to precisely know the exact distance between two convex polyhedral compared to other methods. Instead of performing complex algebraic solution that has been used in various techniques to determine the distance computation algorithm in determining collision, we have proposed a technique that use the efficiency of Axis-Aligned Bounding-Box (AABB) that create boundary surrounding the last piece of triangle in Bounding-Volume Hierarchies (BVH). This technique is called Dynamic Object Point for Distance Computation (DOPDC) technique. It explicitly used the inner and outer AABB capabilities that have been created using dynamic parallel lines. Section II will describe about previous works on collision detection and distance computation algorithms. Section III explained in detailed regarding Bounding-Volume Hierarchies and the theoretical framework and the pseudo code of the algorithm along with the analysis. Section IV conclude our framework.

II. RESEARCH BACKGROUND

Collision detection research studies have been performed well by many top researchers in various fields such as computational geometry, robotics, computer games and animation and computer graphics [3, 8-24]. Some of the main references for other researcher in term of survey papers are [22, 25-32]. Hence in this section, we described few selected

technique that are related to the work of collision detection especially in discrete and continuous collision detection.

III. THEORITICAL FRAMEWORK

In this paper we will describes our theoretical framework of our propose technique that use the capabilities of Axis-Aligned Bounding-Box to calculate the distance between two objects down to the distance between two triangles.

A. Region Creation

At first, from the Bounding-Volume Hierarchies (BVH) that enclosed the triangle down to the last piece of triangle, we will use the AABB for each of the triangle that approximately close to each other. By maintaining one of the axes that has the shortest distance between two AABBs, we can create an invisible AABB just to enclose both of BV into one (combined). By doing this, we can get a new region that has two triangle that might close together to create a contact or determine the contact between them. Figure 1 explain how one of the axes can approximately know which triangles that most likely can come into contact and then determine the distance.

B. Parallel Line Elimination using Outer AABB techniques

Parallel Line Elimination (PLE) stands for elimination of the axes lines that we do not require for distance computation between both triangles. In this case, we need to search for any lines that we do not need for intersection and distance computation testing. Figure 2 explains the situation.

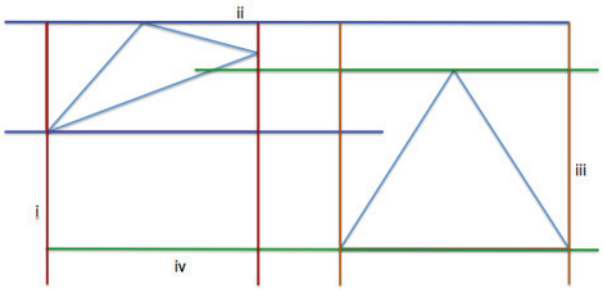

Figure 1 Parallel Line Elimination process to remove any lines that we do not require for nearest intersection contact.

From the Figure 2, we found that using the maximum and minimum point that created AABB, we can do the elimination. Given A_{xmin} and A_{xmax} as minimum and maximum points of first triangle (the leftmost), and B_{xmin} and B_{xmax} for the second triangle (the rightmost). First we will perform comparison between A_{xmin} and B_{xmin} in order to determine which one will be having the greater or lower value. We perform elimination or remove the lower value, as it is not require for distance computation algorithm. This only applies for X-axis where the nearest distance most likely to lies between A_{xmax} and B_{xmin}. Thus in this case A_{xmin} is lower that B_{xmin} and we will select A_{xmin} as candidate to be removed from our distance computation calculation. The same thing applies for maximum points comparison between both triangles in X-axis where between A_{xmax} and B_{xmax}, only B_{xmax} will be removed from our calculation as the B_{xmax} is largest value of all X-axis. In short, between A_{xmin} , A_{xmax} , B_{xmax}, and B_{xmin}, remove any points that has highest and lowest X-axis values thus leaving only the X-axis point that most likely the nearest of X-axis that can come into contact.

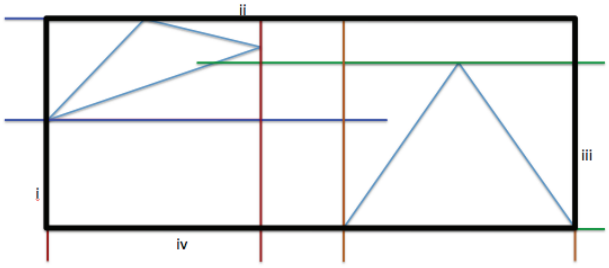

Figure 2 Outer AABB for removing those lines that are not required for Distance Computation Calculation

Outer AABB is one of the techniques that can be used to eliminate those lines that we do not need in order to calculate the distance between those two triangles. By finding the maximum and minimum points of all triangles (that we used to create AABB for each triangle), we can create a new Outer AABB to find any points that belong to Outer AABB creation. In previous paragraph we have explained how to find the lowest and highest value for each axis. However, we can perform faster elimination by performing Outer AABB technique. Figure 3 shows Outer AABB technique.

It works be enclosing the triangle that might come into contact (nearest) with the big AABB. Then, once we have found points that created Outer AABB (big AABB), we can determine which point that most likely to create them. In this case, A_{ymax}, and B_{ymin} are belongs to Outer AABB creation and thus we removed them from our distance computation calculation. Figure 4 shows the final configuration for us to compute the distance.

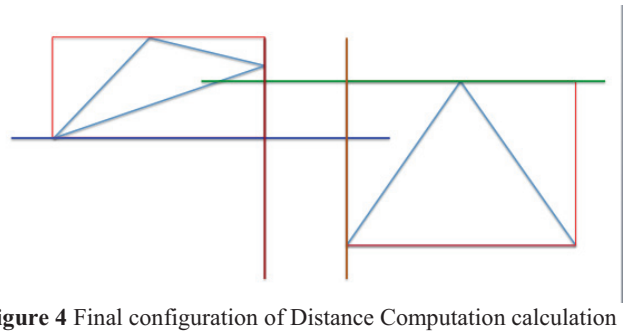

Figure 4 Final configuration of Distance Computation calculation where points/lines that close to only the nearest intersection of area that might come into contact is required

C. Midpoint of AABB

From a closer look at the 'empty box' that lies between the two triangles in AABB, we now can find the midpoint of the AABB as one of the main component for distance computation calculation. We will then transforming the old origin point located at the coordinates (0,0,0) into the new one (let say

$X_{neworigin}$, $Y_{neworigin}$, and $Z_{neworigin}$). Figure 5 explains the procedure to determine a new origin point for distance computation algorithm that we develop.

From Figure 5, a New Origin Point (NOP) can easily be calculated by using the formula below:

$$\frac{B_{xmin} - A_{xmax}}{2} = NOP_{x-mid}$$

the same process will happens to the y coordinate by changing it to:

$$\frac{B_{ymax} - A_{ymin}}{2} = NOP_{y-mid}$$

Once NOP has been calculated, we will save it into the memory for distance computation calculation after this.

D. *Figures and Tables Middle Box of AABB for Intersection Points*

The Middle Box of AABB is an "empty box" that temporarily created to find the midpoint of rectangle in order to initialize NOP. Once the NOP has been saved into memory, the process of distance computation calculation continues with the process of finding the nearest side of triangle that may come into contact. Figure 5 shows the corresponding behavior that need to be check in order to find the nearest side of triangle.

$$y_{edgeleft} = \frac{N_{xmax} - N_{ymin}}{M_{xmax} - M_{ymin}} x + C_A$$

$$y_{rightleft} = \frac{Q_{ymax} - Q_{xmin}}{P_{ymax} - P_{xmin}} x + C_B$$

where C_A *and* C_B is the interception of y axis that need to be calculated first.

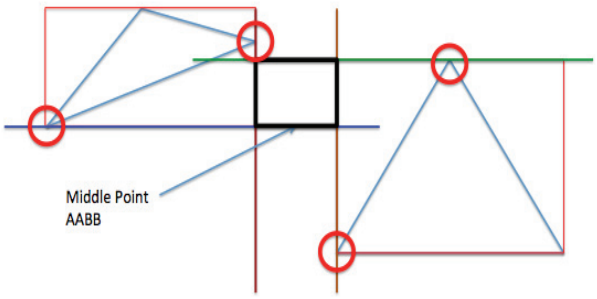

Figure 5 Middle Box of AABB located just between two triangles and contains Middle Point AABB or NOP

From the Figure 5, the process of finding the right edge of intersection area started by using all points that created the Middle Box of AABB. By utilizing the A_{xmax}, A_{ymin}, B_{xmin}, and B_{ymax}, we can easily create an edge and thus it can be declared as the nearest edge that might come into contact. Line equations for both edges must be calculated in order to obtain points or triangle that intersected with a line that is created from the NOP to both edges. From Figure 6, we can obtains two line equations of two points.

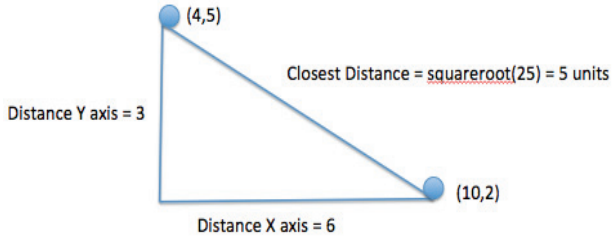

Figure 6 Trigonometric rules of finding the distance between two points

IV. Experimental Works and Analysis

In our experiments, we have conducted two tests that represented the speed of the distance computation and the accuracy of the distance by comparing with all the algorithms. We have loaded the environment with 10 types of triangles that is iterated with different types of triangle. From 10 types of triangle, we iterated for 90 times where each triangle will be tested against another shape of triangle. Our experiments do not involve the same type of triangle in order to maintain the randomization of our algorithm with others. Moreover, it is a rare case where each triangle in certain object will be tested against the same size. For example, if the object contains more than 1 million of triangle in random movement and angle, compute the distance with another object that also might have more than 1 million of triangle, thus, it will have a very minimum percentage that we will be having the same triangle size intersection. Hence in this experiments, we conducted 90 tests (where each triangle will be tested with the other nine thus resulting 90 tests) in order to perform analysis of DyOP, Lin Canny, and GJK algorithm.

In this experiments, we set up the environment by iterating a moving triangle with a static triangle. Each moving triangle will be tested against another shape of triangle. Given the name for each triangle is Obj1 until Obj10, thus we iterated Obj 1 with Obj2, Obj1 with Obj3, Obj1 with Obj4, Obj1 with Obj5, Obj1 with Obj6, Obj1 with Obj7, Obj1 with Obj8, Obj1 with Obj9, and Obj1 with Obj10. Where Obj1 became the moving triangle candidate while others become the static triangle for our experiments. For another iteration, we will repeat the process for Obj2, Obj3 until Obj10 and skipped any same Obj testing. In this case, no such thing as Obj1 with Obj1, Obj2 with Obj2, and others. For all triangles, a total 90 iterations/tests will be conducted. We also setup a fixed distance for all algorithms. Instead of randomization, we need to check the efficiency of the algorithm by maintaining only a single fixed distance. Thus, all the algorithm will be conducted

their test using the same distance check. The time frame is captured for nine iterations for each Obj1 until Obj10. Figure 7 shows our result of this experiment.

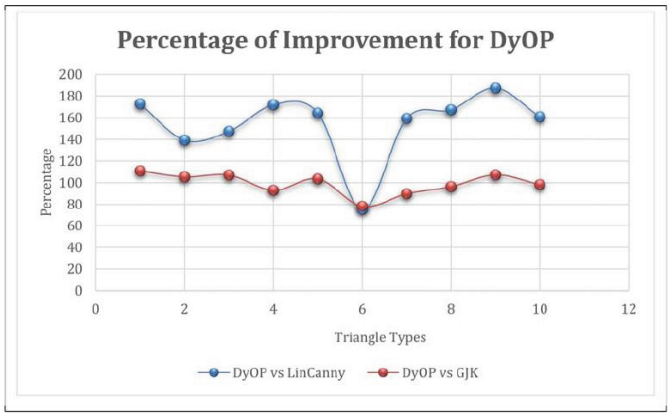

Figure 7 Speed Computation for Distance Computation

Based on figure 7, the highest of percentage between DyOP and LinCanny algorithm is 187.2727273% while the lowest being 75%. This means that our DyOP has proven to provide faster distance computation algorithm other than the other algorithms. We also have a 107.2727273% increase for DyOP versus GJK algorithm and the lowest being 78.125%. This proven enough to us that our algorithm works in better efficiency than the other algorithms.

CONCLUSION AND FUTURE WORK

We have presented our current research progress on development of a novel algorithm of Distance Computation using Axis Aligned Bounding Box (AABB) Parallel Distribution of Dynamic Origin Point. Currently, the experimental process in being carried out using C++ programming language in Visual Studio C++ 2010. Stanford 3D model has been used in our experiments in order to maintain the standard 3D object that had been used for collision detection testing.

Based on all experiments, we concluded that our proposed algorithm of DyOP is more superior to the other two algorithms of GJK and Lin-Canny. The speed of computing distance is increase between ranges of 150% to 180% for certain type of triangles. Although the implementation of GJK and Lin-Canny is following the implementation style of our code is not based on any other codes. But we followed strictly the concept of GJK and Lin-Canny in order to avoid any changes of the method. This is because there is numerous implementation style and code exists for both method but all just follow the same concept with their own style of implementation. However, in this research, all the concept and theoretical aspect of GJK and Lin-Canny is based on the original paper.

Our DyOP algorithm is implemented in 2D environment where it is the easiest part to see the contribution of this algorithm. It is the fundamental idea that we need to chase down to the bottom of complex model, which is a triangle in order to visualize properly what is the idea of DyOP compared to any other methods. Moreover, it is to give other researcher about an overview about this new and novel algorithm at the fundamental level before implement it into any other applications. However, our experiments is just based on distance computation only and not involving any collision response or penetration distance calculation.

ACKNOWLEDGMENT

We would like to thank for Centre for Telecommunication Research and Innovation (CeTRi) for research feedback and Dr. Abdullah Bade and Prof. Datuk Dr. Mohd Harun Abdullah from Universiti Malaysia Sabah for PhD supervision.

REFERENCES

[1] L. Hanwen and W. Yi, "Coherent hierarchical collision detection for clothing animation," in *Haptic Audio Visual Environments and Games (HAVE), 2011 IEEE International Workshop on*, 2011, pp. 129-134.

[2] W. Zhao and L. Wang, "A fast collision detection algorithm suitable for complex virtual environment," in *Transportation, Mechanical, and Electrical Engineering (TMEE), 2011 International Conference on*, 2011, pp. 502-505.

[3] 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, 2008, pp. 486 - 491.

[4] 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.* , 2008, pp. 1834-1839.

[5] M. Haciomeroglu, R. G. Laycock, and A. M. Day, "Dynamically populating large urban environments with ambient virtual humans," *Comput. Animat. Virtual Worlds,* vol. 19, pp. 307-317, 2008.

[6] J. Willmott, L. I. Wright, D. B. Arnold, and A. M. Day, "Rendering of large and complex urban environments for real time heritage reconstructions," presented at the Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage, Glyfada, Greece, 2001.

[7] J. X. Chen, N. d. V. Lobo, C. E. Hughes, and J. M. Moshell, "Real-time fluid simulation in a dynamic virtual environment," *Computer Graphics and Applications, IEEE,* vol. 17, pp. 52-61, 1997.

[8] B. Chen, X. Ye, L. An, and Y. Wang, "Detection of Collision and Self-Collision Using QPSO for Deformable Models," in *Intelligent System Design and Engineering Application (ISDEA), 2012 Second International Conference on*, 2012, pp. 1028-1031.

[9] H. Qu and W. Zhao, "Fast Collision Detection of Space-Time Correlation," in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, 2012, pp. 567-571.

[10] K. Okada, M. Inaba, and H. Inoue, "Real-time and Precise Self Collision Detection System for Humanoid Robots," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1060-1065.

[11] Y. Chun-yan, Y. Dong-yi, W. Ming-Hui, and P. Yun-he, "A new horizontal collision detection scheme for avatar with avatar in collaborative virtual environment," in *Machine Learning and*

Cybernetics, 2005. Proceedings of 2005 International Conference on, 2005, pp. 4961-4966 Vol. 8.

[12] O. Arcila, S. Dinas, and J. M. Banon, "Collision detection model based on Bounding and containing Boxes," in *Informatica (CLEI), 2012 XXXVIII Conferencia Latinoamericana En*, 2012, pp. 1-10.

[13] B. He, Y. Wang, and J. Zhao, "An improved method of continuous collision detection using ellipsoids," in *Computer-Aided Industrial Design & Conceptual Design, 2009. CAID & CD 2009. IEEE 10th International Conference on*, 2009, pp. 2280-2286.

[14] S. Zhang, Z. Jiang, and L. Li, "A Collision Detection Method Based on the Virtual Occluders," in *Computer Science-Technology and Applications, 2009. IFCSTA '09. International Forum on*, 2009, pp. 410-413.

[15] W. Meiping, S. Liyun, and R. Zhijun, "A Hierarchical Collision Detection Algorithm for VA," in *Control and Decision Conference (CCDC), 2010 Chinese*, 2010, pp. 4315-4319.

[16] M. M. Ismail, M. A. Othman, H. A. Sulaiman, M. H. Misran, R. H. Ramlee, A. F. Z. Abidin*, et al.*, "Firefly algorithm for path optimization in PCB holes drilling process," 2012, pp. 110-113.

[17] Z. Wei and L. Lei, "Improved K-DOPs collision detection algorithms based on genetic algorithms," in *Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on*, 2011, pp. 338-341.

[18] S. Yanchun and S. Xingyi, "Research and improvement of collision detection based on oriented bounding box in physics engine," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, 2011, pp. 73-76.

[19] 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.

[20] H. A. Sulaiman, A. Bade, D. Daman, and N. M. Suaib, "Collision Detection using Bounding-Volume Hierarchies in Urban Simulation," presented at the The 5th Postgraduate Annual Research Seminar, Faculty of Computer Science & Information System, UTM, 2009.

[21] 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.

[22] 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.

[23] 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.

[24] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *Robotics and Automation, IEEE Journal of,* vol. 4, pp. 193-203, 1988.

[25] 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.

[26] 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.

[27] 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.

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

[29] M. C. Lin and D. Manocha, "Collision and Proximity Queries," in *In Handbook of Discrete and Computational Geometry, 2nd Ed*. vol. 35, ed Boca Raton, FL: CRC Press LLC, 2004, pp. 787-807.

[30] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha, "Fast continuous collision detection for articulated models," presented at the Proceedings of the ninth ACM symposium on Solid modeling and applications, Genoa, Italy, 2004.

[31] S. A. Gottschalk, "Collision queries using oriented bounding boxes," The University of North Carolina at Chapel Hill, 2000.

[32] G. Zachmann, "Virtual Reality in Assembly Simulation - Collision Detection, Simulation Algorithms, and Interaction Techniques," Department of Computer Science, Darmstadt University of Technology, Germany, 2000.