Australian Journal of Basic and Applied Sciences, 7(10): 490-499, 2013
ISSN 1991-8178

Measuring Distance between Nearly Intersected Objects in Narrow Phase Collision Detection

[1]Hamzah Asyrani Sulaiman, [2]Mohd Azlishah Othman, [3]Ridza Azri Ramlee, [4]Muhammad Harris Misran, [5]Muhammad Noorazlan Shah Zainudin, [6]Muhammad Harris Misran, [7]Abdullah Bin Bade, [8]Mohd Harun Bin Abdullah

[1-6]Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka, Malaysia
[7,8]School of Science and Technology, Universiti Malaysia Sabah, 88440 Kota Kinabalu, Sabah, Malaysia

Abstract: Calculating the distance between two or more object primitives (or triangles) in virtual environment application before collision occurs is important elements in narrow phase collision detection system. Given pairs of nearly colliding objects, their triangles must be checked one by one until the most shortest distance is founded and thus the computation cost for checking collision reduced by checking the nearest triangles that possible to collide. Hence, in this research, we used Heron's formula for calculating the distance between objects that nearly collide and compared it using vector-based calculation. We have found that the formula increased the speed of distance computation slightly faster compared to the vector-based calculation for single triangle checking with minimum memory requirements. In this paper, we explained the procedures of using Heron's Formula and vector-based techniques for computing distance and the experimental results between those two techniques. It is believed that it could help to speed up the process of determine the precise contact between colliding objects while maintaining the accuracy of the collision checking.

Key words: Collision detection, Heron's formula, distance computation, virtual environment.

INTRODUCTION

In virtual environment application, performing collision detection is one of the important elements for computer graphics and visualization research area. Considering application such as computer games and medical simulation, detecting possible area that object most likely to collide is always a continuing problem exists from decades ago. Each collision detection technique develop by programmers and researchers only coupe a certain specific application where the targeting application must consider either to speed up the process or measure the accuracy of the detection. Most computer games at earlier age used collision detection technique that fast enough to give appropriate response to the users or games. However, today computer games concentrate on using real physics and real mathematical calculation in order to increase the realism in the eyes of most users. Hence, it is essential to develop and perform research studies in order to increase the realism of playing computer games and the simulation itself (Sulaiman, H.A., *et al.*, Sulaiman, H.A., *et al.*, 2013; Sulaiman, H.A., *et al.*, 2010; Sulaiman, H.A., *et al.*, 2009; Suaib, N.M., *et al.*, 2009).

Collision detection technique consisting two major parts mainly broad phase collision detection and narrow phase collision. At first, the object is undergoing broad phase collision detection whereas the object is been realized as a simple object that answer the easy question whether they are colliding or not. In most cases, Bounding-Volume (BV) is used for this simple task. More complicated version used Bounding-Volume Hierarchies (BVH) whereas the BV itself is representing as a big BV and then enclosing a smaller BV until the end of the hierarchy (Qu, H. and W. Zhao, 2012; Arcila, O., S. Dinas and J.M. Banon, 2012; Wei, Z. and S. Jing, 2012; Rui, H., 2012; Sulaiman, H.A., *et al.*, 2010; Sulaiman, H.A., *et al.*, 2010; Suaib, N.M., *et al.*, 2009; Sulaiman, H.A., *et al.*, 2009; Nguyen, A., 2006; Klosowski, J.T., *et al.*, 1998). Bounding-Volume Hierarchies of k-DOPs has been proposed by Klosowski *et al* (1998) in 1998 by using a convex polytopes with some orientations of k value. By implemented this technique, their algorithm showed promising results that could benefits in complex static environment with collision detection algorithm. Other researchers also used various of BVs for the BVH such as boxtrees (Okada, K., *et al.*, 2005; Wang, X.P., *et al.*, 2010), Axis-Aligned Bounding-Box (AABBs) (Okada, K., *et al.*, 2010; Zhiwen, Y. and W. Hau-San, 2006; Feixiong, L., *et al.*, 2009; Hanwen, L. and W. Yi, 2011; Gong, J., J. An and L. Cui, 2011; Yi-Si, X., *et al.*, 2010; Tu, C. and L. Yu, 2009; Zhang, X. and Y.J. Kim, 2007; Weller, R.E., *et al.*, 2006), Oriented Bounding-Box (OBB) (Zhiwen, Y. and W. Hau-San, 2006; Feixiong, L., *et al.*, 2009; Tu, C. and L. Yu, 2009; Chun-yan, Y., *et al.*, 2005; Zhao, W. and L. Wang, 2011; Yanchun, S. and S. Xingyi, 2011; Chang, J.W., *et al.*, 2010; Lu, C. and Q. Guofeng, 2010; Zhou, X., 2010; Shen, X.L. and J.S. Zhang, 2010; Chang, J.W., *et al.*, 2009; Gottschalk, S., *et al.*, 1996), k-Dop (Zhang, P. and G.L. Du,

Corresponding Author: Hamzah Asyrani Sulaiman, Faculty of Electronic and Computer Engineering, Universiti Teknikal Malaysia Melaka, 76100 Durian Tunggal, Melaka, Malaysia

2011) Oriented-Dops (Yanchun, S. and S. Xingyi, 2011; Chang, J.W., *et al*., 2009; Bade, A., *et al*., 2006) and convex hulls (Zhang, X., *et al*., 2006; Cameron, S., 1997; Quinlan, S., 1994; Gilbert, E.G. and C.P. Foo, 1990).

Distance Computation Algorithm for collision detection has been studied for the past three decades ago where M. Orlowski (1985) published a paper of "The Computation of the distance between polyhedra in 3-space", E.G. Gilbert, D.W. Johnson, and S.S. Keerthi (1988) published "A fast procedure for computing the distance between objects in three-dimensional space" and M.C. Lin (1991) published a popular paper of "A Fast Algorithm for Incremental Distance Calculation". Based on this paper, distance computation is mainly a method to determine the approximately high precision distance between pair of convex polyhedra. Distance computation algorithm is highly depends on the smallest step of object movement toward another objects as all computer simulation consists of coordination system. Thus, it has been used in another type of collision detection technique that focused on accuracy which is Continuous Collision Detection (CCD). Compared to Discrete Collision Detection (DCD), CCD provides a sequence of small, discrete steps that looks like a continuous movement.

Vector-Based Calculation:

A. Distance between Vertices:

In order to perform vector-based calculation for nearly intersected triangles, first we must recognized the potential individual triangle for each object that most likely to have the nearest possible distance with another object triangle. Once the potential triangle that close to another object triangle has been identify, we will proceed by checking the nearest distance between all vertices of each one triangle to another triangle. Figure 1 shows an example with two-dimensional (2D) triangle that we used to calculate the distance between both triangles. Thus, only vertex-to-vertex and edge-to-vertex testing is required. However, edge-to-edge test is not accounted in this procedure as it produces the same result as vertex to vertex distance if the edge of one triangle is parallel with edge of another triangle.

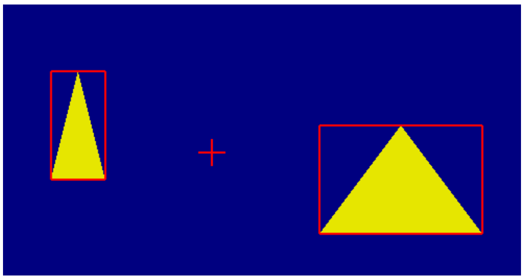


Fig. 1: Two nearly intersected triangles.

From figure 1, each triangle is bounded with an AABB in order to find the Dynamic Origin Point (DyOP) that represented by "+" symbol [Reference]. The DyOP technique is used in order to find the nearest edge or vertices and thus minimizing the efforts of testing all triangles vertices for the nearest distance. Once the DyOP ("+" sign) has been founded, we only need to calculate for the selected vertices and edges that near to the DyOP. Figure 2 illustrates the potential vertices that need to be checked for the distance and the edge-to-edge distance between both triangles in this 2D example.

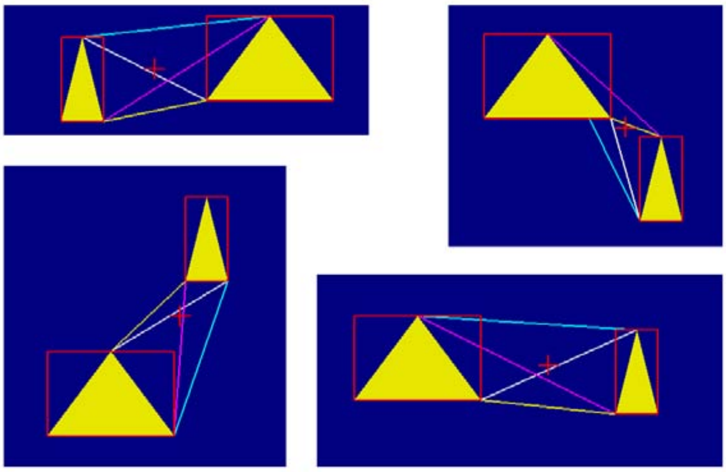


Fig. 2: Nearest possible distance using DyOP technique. Each connected lines (blue, purple, white and yellow line) represented the shortest two vertices towards the DyOP.

Based on the selected vertices and edges using DyOP technique, we need to find the distance between each vertex of first object with the vertices of second object. Let the small triangle denoted as “A_Tri” while the bigger one as “B_Tri” triangle. For A_Tri, two nearest vertices named “A_{vx1}” and “A_{vx2}” while B_Tri has nearest vertices of “B_{vx1}” and “B_{vx2}”. All vertices are stored in vector based format that contains x, y, and z coordinates. The vertices is continuously updated if there is a movement using matrix transformation. Thus, the distance between verticesA_{vx1} and $B_{vx1}$of of A_Tri and B_Tri is depicts in equation 1:

$$Distance\ A_{vx1} with\ B_{vx1} = \sqrt{((A_{vx1}.x - B_{vx1}.x)^2 + (A_{vx1}.y - B_{vx1}.y)^2 + (A_{vx1}.z - B_{vx1}.z)^2)}$$

Eq. 1

The calculation must be done for A_{vx1} with B_{vx2}, A_{vx2} with B_{vx1}, and A_{vx2} with B_{vx2}. Once all the distance between vertices has been calculated, we keep the shortest distance between vertices in the computer memory and continue with edge-to-vertex test.

B. Distance Edge-to-Vertex:

The distance from a vertex to an edge is determined by the perpendicular distance between the vertex and the corresponding edge. Let say $P_o(x_o, y_o, z_o)$ is a vertex point, $\bar{s} = \{m; n; p\}$ is representing an line for corresponding edge and $Q_1(x_1, y_1, z_1)$ is a coordinate on that edge or line $\bar{s}$. Then the distance of P_o and line $\bar{s}$ can be found using the formula below:

$$distance = \frac{|\overline{P_{\bar{o}} Q_{\bar{1}}} \times \bar{s}|}{|\bar{s}|}$$

Eq. 2

Figure 3 illustrates how the distance is found using the formula above.

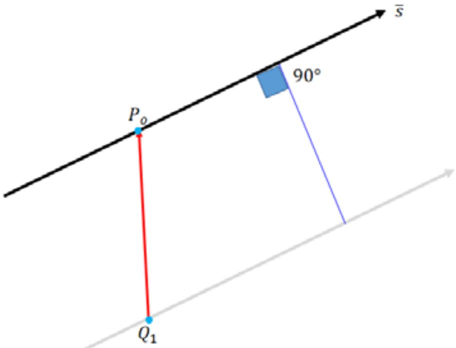


Fig. 3: Edge to vertex distance computation.

In our implementation, since the edge has been determined using DyOP technique, our job is to find the vector for edge of A_{vx1} and A_{vx2} and the vector for edge of B_{vx1} and B_{vx2}. Figure 4 shows the corresponding vector for each triangle. Each edge will be named as A_{edge} and B_{edge} .

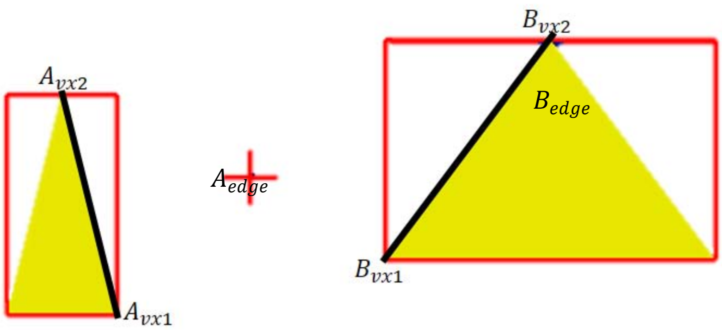


Fig. 4: Black line represented edge for A_Tri and B_Tri.

The vector for both edges can be defined as follow in equation 3 and 4 below:

$$A_{edge}.x = A_{vx2}.x - A_{vx1}.x$$
$$A_{edge}.y = A_{vx2}.y - A_{vx1}.y$$
$$A_{edge}.z = A_{vx2}.z - A_{vx1}.z$$

Eq. 3

$$B_{edge}.x = B_{vx2}.x - B_{vx1}.x$$
$$B_{edge}.y = B_{vx2}.y - B_{vx1}.y$$
$$B_{edge}.z = B_{vx2}.z - B_{vx1}.z$$

Eq. 4

Once the vector for each edge has been calculated, we need to use A_{edge} and B_{edge} as vector $\bar{s}$ in the formula from equation 2. Hence, A_{edge} could be represented as $\bar{A}_{\overline{edge}} = \{A_{edge}.x; A_{edge}.y; A_{edge}.z\}$ and $\bar{B}_{\overline{edge}} = \{B_{edge}.x; B_{edge}.y; B_{edge}.z\}$.

Next, we need to find dot product of A_{vx1} and A_{vx2} with $\bar{B}_{\overline{edge}}$ and B_{vx1} and B_{vx2} with $\bar{A}_{\overline{edge}}$. Since the vertex point at A_{edge} and B_{edge} is known, based on the formula at equation 2, we could find dot product between both vertices. Figure 5 shows the corresponding calculation between edge and vertex between each triangle.

Using A_{vx1} as point on line $\bar{A}_{\overline{edge}}$ with B_{vx1} and B_{vx2} as the required distance from the B_Tri triangle:-

$$\overline{B_{\overline{vx1}}A_{\overline{vx1}}} = \{A_{vx1}.x - B_{vx1}.x; A_{vx1}.y - B_{vx1}.y; A_{vx1}.z - B_{vx1}.z\}$$

Eq. 5

$$\overline{B_{\overline{vx2}}A_{\overline{vx1}}} = \{A_{vx1}.x - B_{vx2}.x; A_{vx1}.y - B_{vx2}.y; A_{vx1}.z - B_{vx2}.z\}$$

Eq. 6

and using B_{vx1} as point on line $\bar{B}_{\overline{edge}}$ with A_{vx1} and A_{vx2} as the required distance from the A_Tri triangle:-

$$\overline{A_{\overline{vx1}}B_{\overline{vx1}}} = \{B_{vx1}.x - A_{vx1}.x; B_{vx1}.y - A_{vx1}.y; B_{vx1}.z - A_{vx1}.z\}$$

Eq. 5

$$\overline{A_{\overline{vx2}}B_{\overline{vx1}}} = \{B_{vx1}.x - A_{vx2}.x; B_{vx1}.y - A_{vx2}.y; B_{vx1}.z - A_{vx2}.z\}$$

Eq. 6

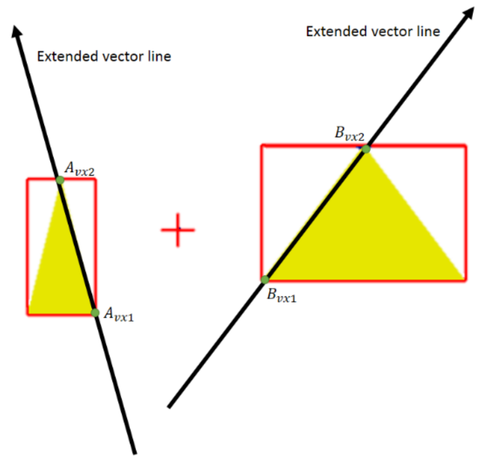


Fig. 5: Based on extended vector line, the perpendicular distance between vertices of corresponding triangle to the edge of triangle will be calculated. However, the calculation will not be done if the vertex fall outside the range of the normal vector line.

In our implementation, the extended vector line is used to calculate the perpendicular distance between each vertex to the edge of another triangle. However, we do not perform any distance computation between vertex and edge if their perpendicular point fall into the outside range of the normal vector line. It is crucial to set up the implementation so that the vertex of corresponding triangle is exactly perpendicular to the edge of vector line. Otherwise, vertex-to-vertex point is consider as the nearest distance.

When the dot product has successfully calculated, we performed cross product with the directing vector of line of corresponding triangle and then find their magnitude. Finally, we need to divide with the directing vector of line magnitude in order to find the distance.

Distance between A_{vx1} withB_{vx1} and B_{vx2} using $\bar{A}_{\overline{edge}}$ as directing vector of line:-

$$distance\ A_{vx1}\ and\ B_{vx1} = \frac{\left|\overline{B_{\overline{vx1}}A_{\overline{vx1}}} \times \bar{A}_{\overline{edge}}\right|}{\left|\bar{A}_{\overline{edge}}\right|}$$

Eq. 7

$$distance\ A_{vx1}\ and\ B_{vx2} = \frac{\left|\overline{B_{\overline{vx2}}A_{\overline{vx1}}} \times \bar{A}_{\overline{edge}}\right|}{\left|\bar{A}_{\overline{edge}}\right|}$$

Eq. 8

and distance between B_{vx1} with A_{vx1} and A_{vx2} using $\bar{B}_{\overline{edge}}$ as directing vector of line:-

$$distance\ B_{vx1}\ and\ A_{vx1} = \frac{\left|\overline{A_{vx1}B_{vx1}} \times \bar{B}_{\overline{edge}}\right|}{\left|\bar{B}_{\overline{edge}}\right|}$$

Eq. 9

$$distance\ B_{vx1}\ and\ A_{vx2} = \frac{\left|\overline{A_{vx2}B_{vx1}} \times \bar{B}_{\overline{edge}}\right|}{\left|\bar{B}_{\overline{edge}}\right|}$$

Eq. 10

Heron's Formula Implementation:

Heron's formula is named after Heron of Alexandria, centuries ago that can be used to calculate area of triangle using a formula below:-

$$T = \sqrt{s(s-a)(s-b)(s-c)}$$

where a,b,c is the length of each edge for the triangle and s is the semiperimeter triangle:-

$$s = \frac{a+b+c}{2}$$

We extend this Heron's formula in order to find the distance between vertex point to the edge of the corresponding triangle by multiplying it by two and divide it by the length between vertex that perform the line for corresponding triangle. Thus the formula becoming:-

$$distance = \frac{\left(\sqrt{s(s-a)(s-b)(s-c)}\right) * 2}{length\ of\ the\ edge}$$

In order to implement Heron's formula into the program, we first find out the corresponding vertex and edge to create a virtual triangle (or temporary triangle region/area). Figure 6 shows one example to find the distance between A_{vx1} and B_{edge} .

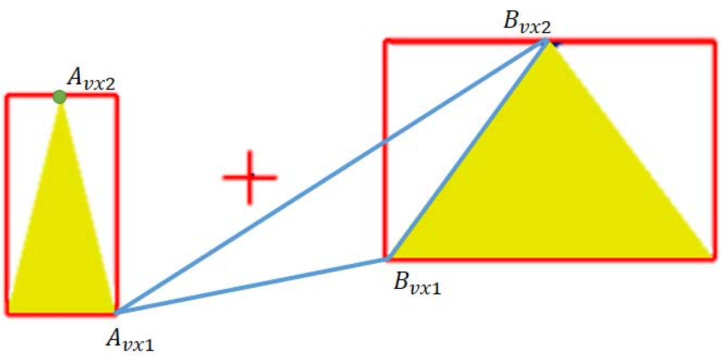


Fig. 6: Virtual triangle or temporary triangle created by using $\boldsymbol{A_{vx1}}$ and $\boldsymbol{B_{edge}}$

From figure 6, let's denote H_{a1b1}, H_{a1b2}, and H_{b1b2} as length for each edge of virtual triangle:-

$$H_{a1b1} = \sqrt{(A_{vx1}.x - B_{vx1}.x)^2 + (A_{vx1}.y - B_{vx1}.y)^2 + (A_{vx1}.z - B_{vx1}.z)^2}$$

Eq. 11

$$H_{a1b2} = \sqrt{(A_{vx1}.x - B_{vx2}.x)^2 + (A_{vx1}.y - B_{vx2}.y)^2 + (A_{vx1}.z - B_{vx2}.z)^2}$$

Eq. 12

$$H_{b1b2} = \sqrt{(B_{vx1}.x - B_{vx2}.x)^2 + (B_{vx1}.y - B_{vx2}.y)^2 + (B_{vx1}.z - B_{vx2}.z)^2}$$

Eq. 13

Based on the equation 11 to 13, the s parameter for all three are:-

$$s = \frac{H_{a1b1} + H_{a1b2} + H_{b1b2}}{2}$$

Eq. 14

and then we put into parameter T, area:-

$$T = \sqrt{s(s - H_{a1b1})(s - H_{a1b2})(s - H_{b1b2})}$$

Eq. 15

The distance between vertex A_{vx1} and B_{edge} can be concluded as follows:-

$$distance = \frac{\left(\sqrt{s(s-H_{a1b1})(s-H_{a1b2})(s-H_{b1b2})}\right)*2}{H_{b1b2}}$$

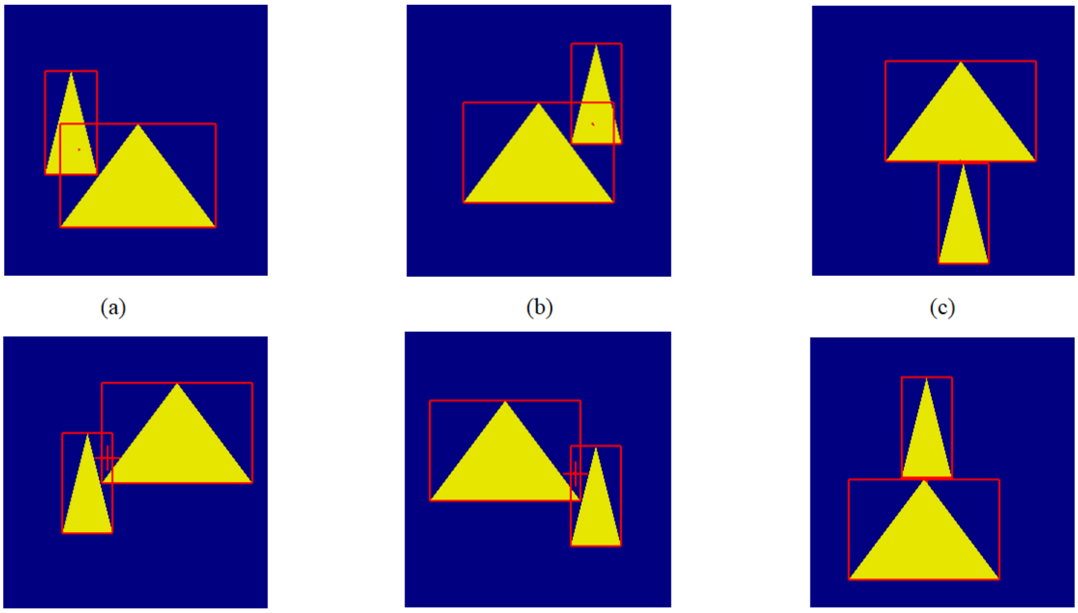
Once all the distance has been found, we only requires to obtain the shortest distance among vertices and edges. In the next section, we described our initial setup for the experiment using 2D triangle with ten different types of triangle. All experiments is undergoing under the same circumstance in order to obtain fine comparison result between vector-based calculation and Heron's formula.

Experiments and Analysis:

The experiments consisting three phases which are the first one is to calibrate the distance between the first triangle and the second triangle with the static movement, the second one is to use rotation, and the last one is to calculate total times to check for distance between all vertices and edges using all ten different types of triangle.

C. Calibrating Distance Between Vector-Based Technique and Heron's Formula:

In order to determine the data that has been provided by the program works perfectly, we perform a calibration testing between the vector-based technique and Heron's formula technique. By selecting two different types of triangle for the calibration, we could determine whether both technique could provide approximately the same distance. The test requires the program to compute nearly intersecting triangle with six different location. From this test, it also helps to recognize the efficiency in calculating distance accurately between vector-based calculation and Heron's formula. All tests is undergone using Windows 8 Operating systems with optimal graphics card in OpenGL environment compiled in Visual C++ 2012 Professional Edition and program is made using C++ language with an OpenGL library. Figure 7 shows the corresponding triangle with appropriate distance and the result is depict in Table 1.



(a) (b) (c)

(d) (e) (f)

Fig. 7: Six possible static nearly intersecting test for calculating the minimum distance between two triangles.

Table 1: Calibration data of six different location of nearly intersected triangle.

	Vector-based distance	Heron's Formula
Figure 5a1 (a)	0.0859907	0.0859835
Figure 5a1 (b)	0.0500157	0.0500232
Figure 5a1 (c)	0.0700004	0.0700134
Figure 5a1 (d)	0.0679097	0.0679067
Figure 5a1 (e)	0.0606312	0.0606284
Figure 5a1 (f)	0.0599999	0.0600008

Based on figure 7, both triangle did not undergone any rotation to switch vertex point and thus it is consider as static triangle testing between two nearly intersecting triangles. From table 1, we could see the calibration data between vector-based distance computation and Heron's Formula. Since the margin is too little between those two techniques, we consider the resulting shows a promising data that both almost show the same exact value. The known possible for the slightly different is might because of the usage of square root for computing distance between those two techniques. In the next sub-section, we measure the random distance between rotated triangles for another six random movement.

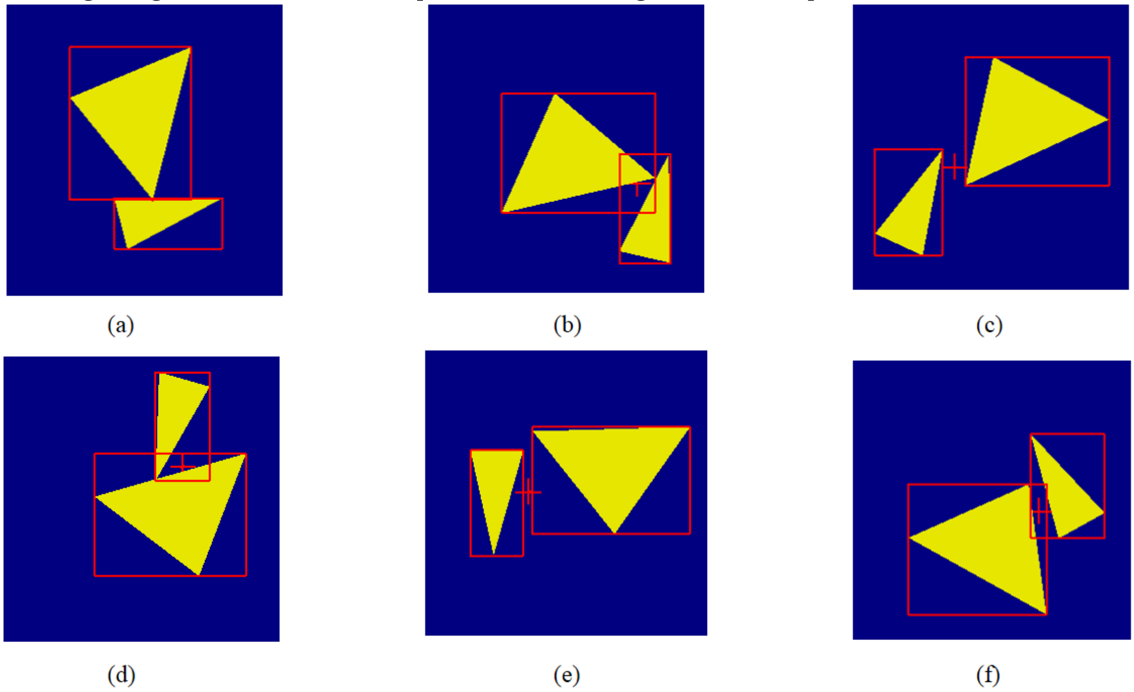
D. Random Distance for Rotated Triangle:

Instead of performing calibration data with static triangle, we conduct an experiment of using rotated triangles and then measure the distance between all vertices of corresponding triangle with another triangle edge. In this case, we run ten possible randomize position to calculate the distance between these nearly intersected triangles. Table 2 shows the data for the distance between those two techniques.

Table 2: Calibration data of six different location of nearly intersected triangle.

	Vector-based distance	Heron's Formula
Random 1	0.9873790	0.9873830
Random 2	0.0895392	0.0895405
Random 3	0.0928680	0.0928632
Random 4	0.0760477	0.0763010
Random 5	0.0414124	0.0414956
Random 6	0.0926698	0.0926732
Random 7	0.0551529	0.0551470
Random 8	0.0754026	0.0753153
Random 9	0.00175346	0.00241474
Random 10	0.00999012	0.00902168

Based on randomize rotation that have been done by the program, we captured several distance for illustration purpose in order to detect any possible error that might occurs if logic error is found in the program. From the data captured, it seems that both achieved almost the same data value up to three to four floating points. Thus, we consider that both method can be used for calculating the distance between vertex and the edge of the triangle. Figure 8 shows randomize position when undergone rotation experiments.



(a) (b) (c) (d) (e) (f)

Fig. 8: Six randomize nearly intersecting test for calculating the minimum distance between two triangles.

E. Total Times for Distance Computation for 10 Randomize Sizes and Types of Triangle:

Final testing involving checking for distance between ten randomize sizes and types of triangle in static triangle mode. Each triangle will be tested against another nine triangle and repeated for another nine triangle. Thus, a total of 90 tests for different sizes and types of triangle and never be tested against the own size and type of triangle. Figure 9 shows the corresponding graph for the experiments.

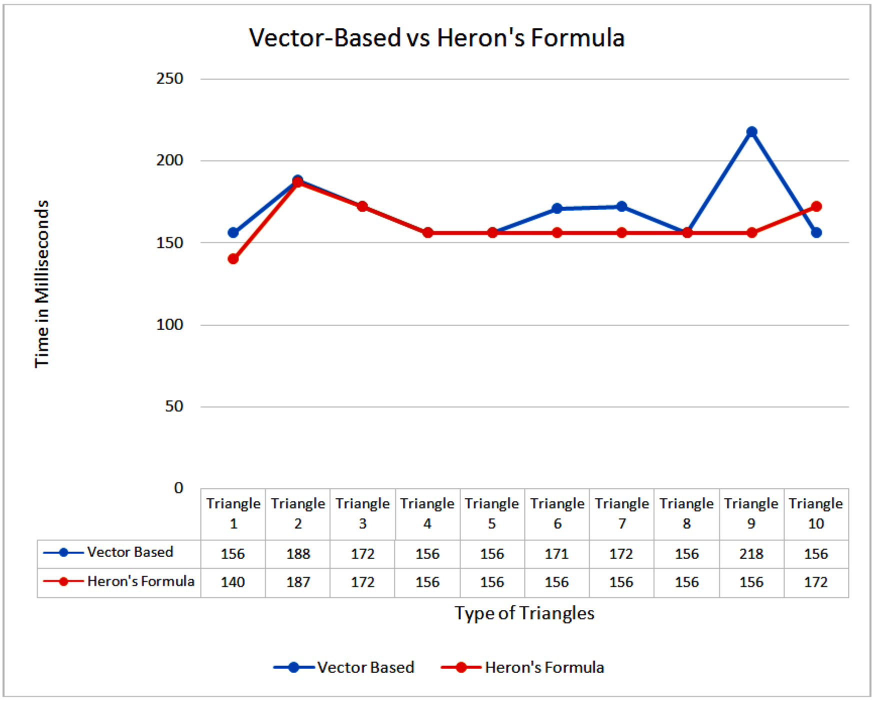


Fig. 9: Vector-Based versus Heron's Formula for Distance Computation

From the figure, the data shows that Heron's Formula is slightly faster than the vector-based method. Since the triangle involves in this experiments consisting ten types of triangles, thus total times per milliseconds for all 90 tests are 1701 milliseconds for vector-based and 1607 milliseconds for Heron's Formula. Thus, it is more than 5% increment of speed using Heron's Formula compared to the vector-based for just ten types of triangle. In virtual environment world, most of applications consisting more than thousands or millions of triangles for calculation where every small increment lead to a lot of increase speed for distance computation.

Conclusion and Future Work:

As a conclusion, our research shows premilinary investigation result based on initial experiment setup in order to show the differences between vector-based technique and Heron's formula in computing distance between nearly intersected triangles. Even though there is only slightly increase of speed for Heron's formula, it still does not yet tested with high complexity objects that might contains more than hundreds to thousands of triangles. Our work is still an ongoing research concentrating on creating a new narrow phase collision detection system starting with distance computation algorithm, determine the precise point of contact for intersecting triangles and calculating penetration depth between intersected triangles.

ACKNOWLEDGMENT

We would like to thanks to Universiti Teknikal Malaysia Melaka for give support and commitment to this research projects especially to Universiti Malaysia Sabah for giving support and research advice through direct and indirect supervision.

REFERENCES

Sulaiman, H.A., M.A. Othman, M.M. Ismail, M.H. Misran, M.A.B.M. Said, R.A. Ramlee, *et al.*, "Quad separation algorithm for bounding-volume hierarchies construction in virtual environment application," *Journal of Next Generation Information Technology,* 4: 63-73.

Sulaiman, H.A., M.A. Othman, M.M. Ismail, M.A. Meor Said, A. Ramlee, M.H. Misran, *et al.*, 2013. "Distance computation using Axis Aligned Bounding Box (AABB) parallel distribution of dynamic origin point,".

Sulaiman, H.A., A. Bade and N.M. Suaib, 2010. "Fast traversal algorithm for detecting object interference using hierarchical representation between rigid bodies," pp: 7-11.

Sulaiman, H.A., A. Bade and N.M. Suaib, 2009."Bounding-volume hierarchies technique for detecting object interference in urban environment simulation," pp: 436-440.

Suaib, N.M., A. Bade, D. Mohamad and H.A. Sulaiman, 2009. "On faster bounding volume hierarchy construction for avatar collision detection,", pp: 430-434.

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

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

Wei, Z. and S. Jing, 2012. "Collision Detection Research for Deformable Objects," in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, pp: 557-561.

Rui, H., 2012. "Optimizing collision detection in 3D games with model attribute and Bounding Boxes," in *Electrical & Electronics Engineering (EEESYM), 2012 IEEE Symposium on*, pp: 589-591.

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

Sulaiman, H.A., A. Bade, D. Daman and N.M. Suaib, 2009. "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.

Suaib, N.M., A. Bade, D. Daman and H.A. Sulaiman, 2009. "Bounding Volume Hierarchy For Avatar Collision Detection: Design Considerations," presented at the The 5th Postgraduate Annual Research Seminar, Faculty of Computer Science & Information System, UTM.

Sulaiman, H.A., A. Bade and N.M. Suaib, 2009. "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.

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

Klosowski, J.T., M. Held, J.S.B. Mitchell, H. Sowizral and K. Zikan, 1998. "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," *IEEE Transactions on Visualization and Computer Graphics*, 4: 21-36.

Okada, K., M. Inaba and H. Inoue, 2005. "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*, pp: 1060-1065.

Wang, X.P., X.C. Dong, H.B. Wang, H.C. Zheng, and S.F. Gu, 2010. "Solution to collision detection based on bounding box in Vega Prime," in *Future Information Technology and Management Engineering (FITME), 2010 International Conference on*, pp: 113-116.

Zhiwen, Y. and W. Hau-San, 2006. "GPCD: Grid-based Predictive Collision Detection for Large-scale Environments in Computer Games," in *Multimedia and Expo, 2006 IEEE International Conference on*, pp: 1025-1028.

Feixiong, L., Z. Ershun, H. Yuefeng, G. Hui and C. Junlai, 2009. "Real-time collision detection and response in virtual global terrain environments," in *Computer-Aided Industrial Design & Conceptual Design, 2009. CAID & CD 2009. IEEE 10th International Conference on*, pp: 2257-2262.

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

Gong, J., J. An and L. Cui, 2011. "Research and Application for Collision Detection Algorithm in Virtools," in *Business Computing and Global Informatization (BCGIN), 2011 International Conference on*, pp: 457-460.

Yi-Si, X., X.P. Liu and X. Shao-Ping, 2010. "Efficient collision detection based on AABB trees and sort algorithm," in *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pp: 328-332.

Tu, C. and L. Yu, 2009. "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.

Zhang, X. and Y.J. Kim, 2007. "Interactive Collision Detection for Deformable Models Using Streaming AABBs," *IEEE Transactions on Visualization and Computer Graphics*, 13: 318-329.

Weller, R.E., J. Klein and G. Zachmann, 2006. "A Model for the Expected Running Time of Collision Detection using AABB Trees," in *Eurographics Symposium on Virtual Environments (EGVE)*, Lisbon, Portugal.

Chun-yan, Y., Y. Dong-yi, W. Ming-Hui and P. Yun-he, 2005. "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*, 8: 4961-4966.

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

Yanchun, S. and S. Xingyi, 2011. "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*, pp: 73-76.

Chang, J.W., W. Wang and M.S. Kim, 2010. "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," *Computer-Aided Design,* 42: 50-57.

Lu, C. and Q. Guofeng, 2010. "Optimization of the collision detection technology in 3D skeleton animation," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, V10-539-V10-543.

Zhou, X., 2010. "Research of collision detection based on OBB in skinned mesh," in *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, pp: V6-643-V6-645.

Shen, X.L. and J.S. Zhang, 2010. "Research of collision detection algorithm based on particle swarm optimization," in *Computer Design and Applications (ICCDA), 2010 International Conference on*, pp: V1-60-V1-63.

Chang, J.W., W. Wang and M.S. Kim, 2009. "Efficient collision detection using a dual OBB-sphere bounding volume hierarchy," *Computer-Aided Design*, vol. In Press, Corrected Proof.

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

Zhang, P. and G.L. Du, 2011. "A fast continuous collision detection algorithm based on K_DOPs," in *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pp: 617-621.

Bade, A., N. Suaib, M.Z.A. and T.S.T.M., 2006. "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.

Zhang, X., M. Lee and Y.J. Kim, 2006. "Interactive continuous collision detection for non-convex polyhedra," *Vis. Comput.,* 22: 749-760.

Cameron, S., 1997. "A comparison of two fast algorithms for computing the distance between convex polyhedra," *Robotics and Automation, IEEE Transactions on,* 13: 915-920.

Quinlan, S., 1994. "Efficient distance computation between non-convex objects," in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on,* pp: 3324 - 3329.

Gilbert, E.G. and C.P. Foo, 1990. "Computing the distance between general convex objects in three-dimensional space," *Robotics and Automation, IEEE Transactions on,* 6: 53-61.