Jurnal
Teknologi

Full Paper

INNER AABB FOR DISTANCE COMPUTATION IN COLLISION DETECTION

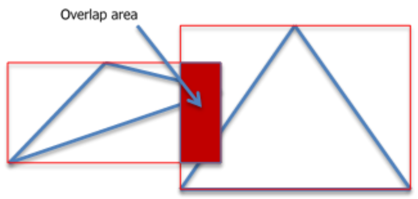
Hamzah Asyrani Sulaiman[a*], Abdullah Bade[b], Mohd Harun Abdullah[b]

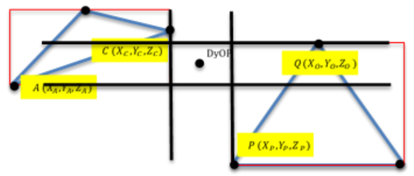
[a]Universiti Teknikal Malaysia Melaka, Melaka, Malaysia
[b]Universiti Malaysia Sabah, Sabah, Malaysia

Article history
Received
15 June 2015
Received in revised form
1 October 2015
Accepted
13 October 2015

*Corresponding author
asyrani@computer.org

Graphical abstract





Abstract

Distance computation technique is one of the important elements for completing narrow phase collision detection system. Checking an accurate distance between two piece of polygons (or some researchers named it as primitives/triangles) is always a challenging tasks where it involves two common measurements, which is speed of the distance checking and the accuracy of the distance itself. In this paper, we performed an experiment using our latest technique called Inner AABB of Dynamic Pivot Point (DyOP) where it's tremendously reduced number of testing and increase the speed of the distance computation. Based on the analyzed results, we believed that our technique is superior compared to other techniques in term of the speed of the detection.

Keywords: Collision detection, virtual environment, distance computation

Abstrak

Teknik pengiraan jarak adalah salah satu elemen penting bagi menyiapkan fasa sempit sistem pengesanan perlanggaran. Semakan jarak yang tepat antara dua sekeping poligon (atau beberapa penyelidik menamakannya sebagai primitif/segi tiga) sentiasa tugas yang mencabar di mana ia melibatkan dua ukuran yang sama, iaitu kelajuan penyemakan jarak dan ketepatan jarak itu sendiri. Dalam kertas ini, kami melakukan satu eksperimen dengan menggunakan teknik terkini yang dikenali sebagai Inner AABB Dinamik Pivot Point (DyOP) di mana ia dengan ketara mengurangkan bilangan ujian dan meningkatkan kelajuan pengiraan jarak. Berdasarkan keputusan dianalisis, kami percaya bahawa teknik kami adalah lebih baik berbanding teknik lain dari segi kelajuan pengesanan.

Kata kunci: Pengesanan perlanggaran, persekitaran maya, pengiraan jarak

1.0 INTRODUCTION

Before an event of collision takes place, both objects that might come into contact need to check their movement into each other and approximate the distance between them [1-8]. By using information such as object directions, nearest point of contact between objects and maximum and minimum points of objects, we can approximate the time of contact and the distance between object before they are colliding. Distance computation is one of the important element for narrow phase collision detection method where it is generally used for medical and other accuracy based simulation. It helps researchers to properly measure the proper distance. Figure 1 shows an overlap area where the contact between both AABBs show a collision between them but in reality, both triangles are not in contact.

The conventional technique tries to find the correct distance from both triangles by computing each vertex, edge, and face. However, the total cost of finding hundreds of them is too expensive and thus it requires a lot of computational cost. Hence, we have implemented in this paper a foundation work on how to prepare a vertex distance computation using Dynamic Origin Point [9, 10] for 3D object without using expensive calculations.

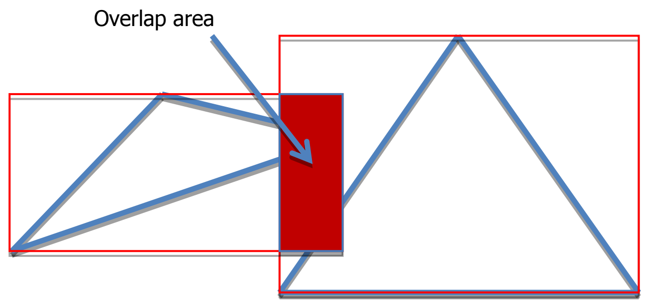


Figure 1 Overlap area for AABB Testing

2.0 IMPLEMENTATION

In a two-component gel, it is easy to modify the molecular structure of either of the two components. In 3D implementation, the algorithm needs to use all X,Y, and Z axis of each vertex in order to find the correct distance. Figure 2 shows an algorithm used for searching the closest midpoint of object face with the DyOP.

```
For each object face,i
        For each triangle vertex in object face,j
                Get the closest midpoint to the DyOP
                Update GetMidDist with nearest distance
                Update getobjTri_MidFace as the closest
point
End Loop
```

Figure 2 3D Distance Computation Main Algorithm for DyOP

Based on the figure 2, for each 3D object that contained face with the triangle vertices, we need to obtain the closest midpoint to the DyOP. The algorithm will perform iteration for all the object faces with their triangle vertices until the closest midpoint is found. The distance between closest midpoints of the corresponding object with the DyOP for each iteration can be calculated using formula (1) below:

$$GetMidDist = \sqrt{X_{Mid}{}^2 + Y_{Mid}{}^2 + Z_{Mid}{}^2} \qquad (1)$$

where X_Mid, Y_Mid, and Z_Mid represented the current midpoint for the object face

Implementation of 3D object distance computation required face to face, face to vertex and face to edge calculation in order to determine the correct distance between a closest point to the nearest intersecting point. By referring to the Figure 3 and making assumption that vertex A and C from left triangle and vertex P and Q from the right triangle near to the DyOP, we calculate the distance between based on formula (2) and we obtained formula (3), (4), (5) and (6).

$$Distance = sqrt[(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2] \qquad (2)$$

Step 1: Finding vertex distance using formula 3

$$Distance = sqrt[(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2 + (Z_{max} - Z_{min})^2] \qquad (3)$$

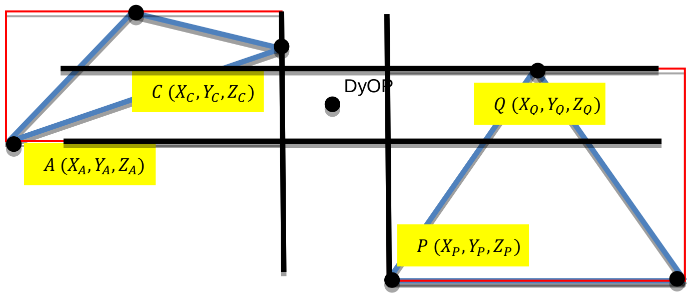


Figure 3 Vertex of A and C from the left triangle and vertex of P and Q from the right triangle are the closest point and edge to the DyOP

$A\ (X_A, Y_A, Z_A)$ with $P\ (X_P, Y_P, Z_P)$

$$Distance\ AP = sqrt[(X_A - X_P)^2 + (Y_A - Y_P)^2 + (Z_A - Z_P)^2] \qquad (4)$$

$A\ (X_A, Y_A, Z_A)$ with $Q\ (X_Q, Y_Q, Z_Q)$

$$Distance\ AQ = sqrt\left[(X_A - X_Q)^2 + (Y_A - Y_Q)^2 + (Z_A - Z_Q)^2\right] \qquad (5)$$

$C\ (X_C, Y_C, Z_C)$ with $P\ (X_P, Y_P, Z_P)$

$$Distance\ CP = sqrt[(X_C - X_P)^2 + (Y_C - Y_P)^2 + (Z_C - Z_P)^2] \qquad (6)$$

$C\ (X_C, Y_C, Z_C)$ with $Q\ (X_Q, Y_Q, Z_Q)$

$$Distance\ CQ = sqrt\left[(X_C - X_Q)^2 + (Y_C - Y_Q)^2 + (Z_C - Z_Q)^2\right] \qquad (7)$$

Step 2: Finding distance between edges by obtaining directing vector for AC and PQ

$$DirectVectABC = Vertex_C - Vertex_A \qquad (8)$$

$$DirectVectPQR = Vertex_Q - Vertex_P \qquad (9)$$

where DirectVecABC representing the left triangle directing vector and DirectVectPQR representing the right triangle directing vector.

Step 3: Find the directing vector between vertices and edges

$$DiVectA\_P = Vertex_A - Vertex_P \quad (10)$$
$$DiVectA\_Q = Vertex_A - Vertex_Q \quad (11)$$
$$DiVectC\_P = Vertex_C - Vertex_P \quad (12)$$
$$DiVectC\_Q = Vertex_C - Vertex_Q \quad (13)$$

Step 4: Distance calculation using cross product

$$DistPointA_p = \sqrt{\frac{X^2_{DiVectA_P} + Y^2_{DiVectA_P} + Z^2_{DiVectA_P}}{X^2_{DirectVectA} + Y^2_{DirectVectA} + Z^2_{DirectVectA}}} \quad (14)$$

$$DistPointA_Q = \sqrt{\frac{X^2_{DiVectA_P} + Y^2_{DiVectA_P} + Z^2_{DiVectA_P}}{X^2_{DirectVectA} + Y^2_{DirectVectA} + Z^2_{DirectVectA}}} \quad (15)$$

$$DistPointC_p = \sqrt{\frac{X^2_{DiVectA_P} + Y^2_{DiVectA_P} + Z^2_{DiVectA_P}}{X^2_{DirectVectA} + Y^2_{DirectVectA} + Z^2_{DirectVectA}}} \quad (16)$$

$$DistPointC_Q = \sqrt{\frac{X^2_{DiVectA_P} + Y^2_{DiVectA_P} + Z^2_{DiVectA_P}}{X^2_{DirectVectA} + Y^2_{DirectVectA} + Z^2_{DirectVectA}}} \quad (17)$$

```
For each object, j
    Step 1: Get two vectors of corresponding triangle with one fixed vertex
    Step 2: Compute normal vector based on both vectors
    Step 3: Normalize the normal vector
        For each triangle vertex, i
            Step 3.1: Get a vector from i to j
            Step 3.2: Compute Magnitude for 3.1
            Step 3.3: Find angle of the vector based on j and I
            Step 3.4: Compute the length of the vector
End Loop
```

Figure 4 3D Distance Computation Main DyOP algorithm

In order to find the distance between faces from vertices of both objects, we have implemented vector-based calculation based on DyOP in order to minimize the computation cost for computing distance between vertices and faces unlike the Lin-Canny method that needs to use brute force technique by using all the triangles in their computation [3-6]. In our implementation, we concentrated on preparing vertex to face calculation as most researchers have made assumption that face to face and edge to face calculation is rarely occurred [3-6, 8, 11-14]. Figure 4 shows main algorithm for vector-based distance between face calculations.

Based on Figure 4, the algorithm starts by finding the two vectors for the targeting triangle with a fixed vertex (let say Triangle PQR and we can use any of the vertex as a reference point). Then, use any of two vertices from another triangle (let say Triangle ABC) to find the directing vectors.

Step 1: Directing vectors based on triangle PQR by using two vertices from triangle ABC

$$DV_{PQ} = Vertex_P - Vertex_Q \quad (18)$$
$$DV_{PR} = Vertex_R - Vertex_P \quad (19)$$

where DV_{PQ} representing the vector calculated using vertex P to vertex Q of triangle PQR and DV_{PR} representing the vector calculated using vertex P to vertex R.

Step 2: Vector normal based on two vectors by computing their cross product.

$$\begin{vmatrix} i & j & k \\ X_{DV_{PQ}} & Y_{DV_{PQ}} & Z_{DV_{PQ}} \\ X_{DV_{PR}} & Y_{DV_{PR}} & Z_{DV_{PR}} \end{vmatrix}$$

$$X_{NormalDV_P} = \left[\left(Y_{DV_{PQ}} * Z_{DV_{PR}}\right) - \left(Y_{DV_{PR}} * Z_{DV_{PQ}}\right)\right] i \quad (20)$$

$$Y_{NormalDV_P} = -\left[\left(X_{DV_{PQ}} * Z_{DV_{PR}}\right) - \left(X_{DV_{PR}} * Z_{DV_{PQ}}\right)\right] j \quad (21)$$

$$Z_{NormalDV_P} = \left[\left(X_{DV_{PQ}} * Y_{DV_{PR}}\right) - \left(X_{DV_{PR}} * Y_{DV_{PQ}}\right)\right] k \quad (22)$$

where the normal vector is represented by $\left(X_{NormalDV_P}, Y_{NormalDV_P}, Z_{NormalDV_P}\right)$

Step 3: Normalize the normal vector and find their magnitude.

Magnitude of the normal vector:

$$MagNormal_{DV_p} = \sqrt{X^2_{NormalDV_P} + Y^2_{NormalDV_P} + Z^2_{NormalDV_P}} \quad (23)$$

Normalize the vector:

$$DV_{Normalize} = \frac{X_{NormalDV_P}}{MagNormal_{DV_p}} \quad (24)$$

Calculate the magnitude of normalized vector:

$$MagDV_{Normalize} = \sqrt{X^2_{DV_{Normalize}} + Y^2_{DV_{Normalize}} + Z^2_{DV_{Normalize}}} \quad (25)$$

Next, we need to calculate the distance between the vectors by performing iteration for each triangle vertex. For example, Triangle ABC has three vertices where each vertex needs to find their directing vector from the triangle PQR reference point. In this case, vertex P from triangle PQR has become a reference point to calculate directing vector towards triangle ABC.

Step 3.1: Get a directing vector for each vertex Triangle ABC to Triangle PQR

$$DV_{PA} = Vertex_P - Vertex_A \quad (26)$$
$$DV_{PB} = Vertex_P - Vertex_B \quad (27)$$
$$DV_{PC} = Vertex_P - Vertex_C \quad (28)$$

Step 3.2: Find the magnitude for each directing vector

$$MagDV_{PA} = \sqrt{X^2_{DV_{PA}} + Y^2_{DV_{PA}} + Z^2_{DV_{PA}}} \quad (29)$$
$$MagDV_{PB} = \sqrt{X^2_{DV_{PB}} + Y^2_{DV_{PB}} + Z^2_{DV_{PB}}} \quad (30)$$
$$MagDV_{PC} = \sqrt{X^2_{DV_{PC}} + Y^2_{DV_{PC}} + Z^2_{DV_{PC}}} \quad (31)$$

Step 3.3: Find angle of the vector based on vector $DV_{Normalize}$ with DV_{PA}, DV_{PB}, and DV_{PC}

$$DV_{PAcos} = \frac{\left[\begin{array}{c}\left(X_{DV_{PA}} * X_{DV_{Normalize}}\right) + \left(Y_{DV_{PA}} * Y_{DV_{Normalize}}\right) \\ +\left(Z_{DV_{PA}} * Z_{DV_{Normalize}}\right)\end{array}\right]}{MagNormal_{DV_p} * MagDV_{PA}} \quad (32)$$

$$DV_{PBcos} = \frac{\left[\begin{array}{c}\left(X_{DV_{PB}} * X_{DV_{Normalize}}\right) + \left(Y_{DV_{PB}} * Y_{DV_{Normalize}}\right) \\ +\left(Z_{DV_{PB}} * Z_{DV_{Normalize}}\right)\end{array}\right]}{MagNormal_{DV_p} * MagDV_{PC}} \quad (33)$$

$$DV_{PCcos} = \frac{\left[\begin{array}{c}\left(X_{DV_{PC}} * X_{DV_{Normalize}}\right) + \left(Y_{DV_{PC}} * Y_{DV_{Normalize}}\right) \\ +\left(Z_{DV_{PC}} * Z_{DV_{Normalize}}\right)\end{array}\right]}{MagNormal_{DV_p} * MagDV_{PC}} \quad (34)$$

Since the result obtained from equation 32 until 34 are in cosine mode, we need to change into the degrees mode by performing anti-cosine (acos function in C++) then multiply with pi/180:

$$\theta_{PAangle} = \left(DV^{-1}_{PAcos} * \frac{180}{\pi}\right)^{o} \quad (35)$$

$$\theta_{PBangle} = \left(DV^{-1}_{PBcos} * \frac{180}{\pi}\right)^{o} \quad (36)$$

$$\theta_{PCangle} = \left(DV^{-1}_{PCcos} * \frac{180}{\pi}\right)^{o} \quad (37)$$

Step 3.4: Total length of each directing vector from Triangle ABC to Triangle PQR

$$Length_{PA} = \left(\sqrt{(X_{vA} - X_{vP})^2 + (Y_{vA} - Y_{vP})^2 + (Z_{vA} - Z_{vP})^2}\right) * DV_{PAcos} \quad (38)$$

$$Length_{PB} = \left(\sqrt{(X_{vB} - X_{vP})^2 + (Y_{vB} - Y_{vP})^2 + (Z_{vB} - Z_{vP})^2}\right) * DV_{PBcos} \quad (39)$$

$$Length_{PC} = \left(\sqrt{(X_{vC} - X_{vP})^2 + (Y_{vC} - Y_{vP})^2 + (Z_{vC} - Z_{vP})^2}\right) * DV_{PCcos} \quad (40)$$

where (X_{vA}, Y_{vA}, Z_{vA}), (X_{vB}, Y_{vB}, Z_{vB}), (X_{vC}, Y_{vC}, Z_{vC}), and (X_{vP}, Y_{vP}, Z_{vP}) representing the vertex point in X,Y, and Z coordinates.

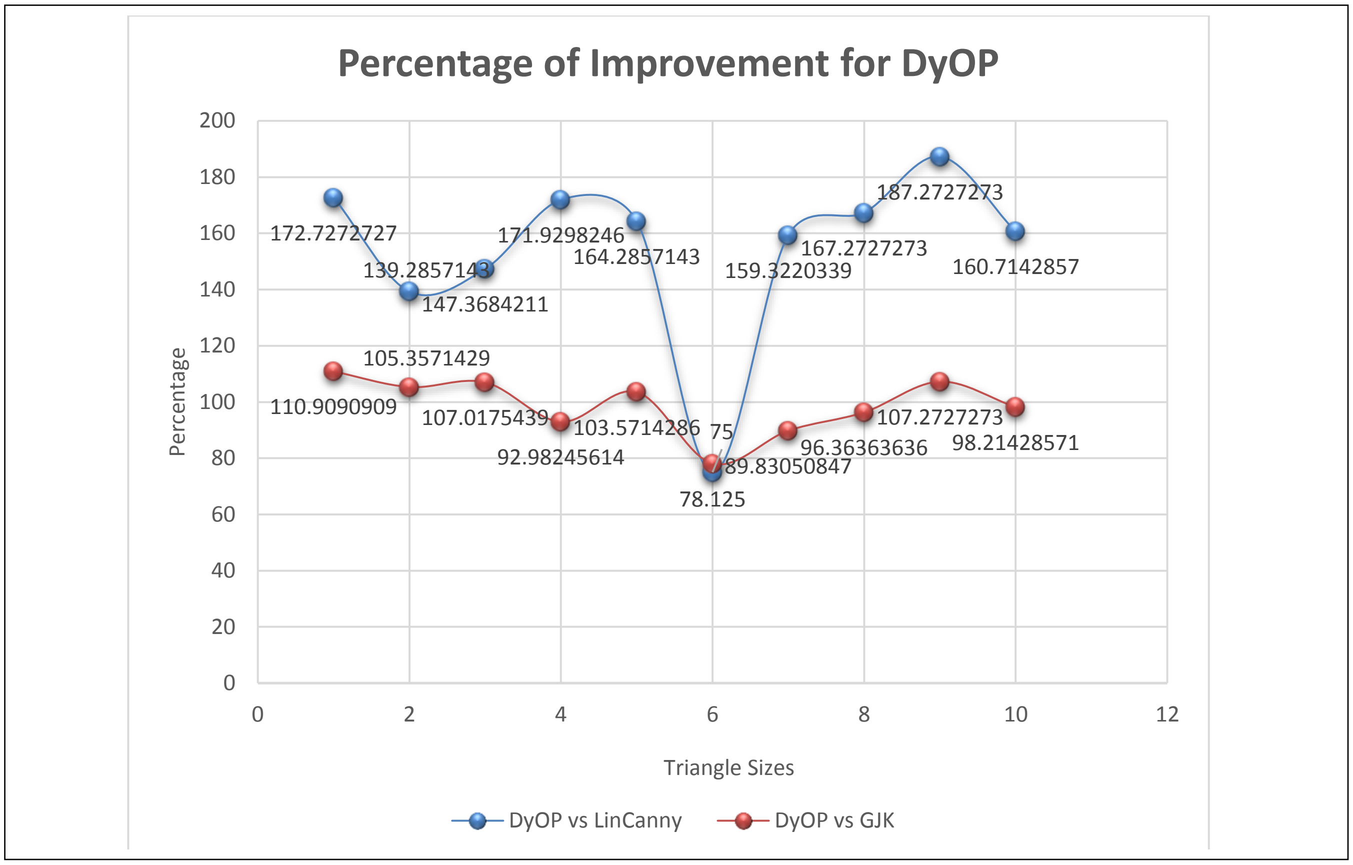


Figure 5 Overall Speed in Percentage for Distance Computation

We believed that the foundation of this algorithm is to work perfectly with our DyOP algorithm for computing distance between two or more primitives. Instead of using all triangles and searching for nearest closest distance, it helps to reduce the complexity and time consumption in finding the distance.

3.0 RESULTS AND DISCUSSION

Based on the graph, DyOP performance is superior compared other two prominent techniques in term of technique speed for distance computation checks. All techniques used the same fixed distance and the same experiment procedures. By improving the speed of distance computation technique, we can potentially increase the speed of collision detection testing especially the narrow phase collision detection types where distance computation technique is mainly used in high accuracy application such as medical simulation and high precision simulation. Which means, more testing will be conducted when the speed is improved and maintaining the accuracy of the collision detection technique. Figure 5 shows in details about the differences in percentage for distance computation speed experiments.

In Figure 5, the highest percentage of speed increases between DyOP and LinCanny technique is 187.3% while the lowest being 75%. Meanwhile, DyOP had shown that it performs better than GJK technique where DyOP set 110.9% highest speed increase and 78.1% for the lowest speed increase. It means that the DyOP has proven to provide faster distance computation technique as compared to Lin Canny and GJK technique. This proven enough that the proposed technique works in better efficiency than the two techniques.

4.0 CONCLUSION

Considering all experiments conducted, we conclude that the proposed technique DyOP is superior as compared to GJK and Lin-Canny techniques. In term of distance computation, the computational speed increase with the almost same accuracy range 150% to 180% for some triangles. Our justification is proved from the testing where number of vertices, edges or faces, the speed for narrow phase collision detection checking are significantly reduced. Both GJK and Lin-Canny implementation are strictly followed it original research paper.

Meanwhile for 3D virtual environment experiment, we have successfully implemented the proposed technique (DyOP) and perform benchmarking comparison between DyOP and Lin-Canny technique. We stressed out that DyOP technique is capable to withstand in any environment and conditions including translational and rotational movement. Instead of performing two-phase collision checking, DyOP can perform collision checking with only just a single phase with optimal accuracy and fast response result. Hence, with this results, we have answered the first and second objectives which are to produce efficient and high performance technique for narrow phase collision detection.

Acknowledgement

The authors would like to thank Universiti Teknikal Malaysia Melaka and Ministry of Higher Education Malaysia for the financial support given through the research grant number FRGS (RACE)/2013/FKEKK/TK3/4/F00197 with a title of A Novel Interference Detection Algorithm Using Planar Subdivision of Axis-Aligned Bounding-Box for Virtual Reality Application.

References

[1] Sulaiman, H. A., Othman, M. A., Salahuddin, L., Zainudin, M. N. S., Salim, S. I. M., Ismail, M. M. *et al.* 2013. Distance Approximation Using Pivot Point In Narrow Phase Collision Detection. *Proc. of 2013 3rd Int. Conf. on Instrumentation, Communications, Information Technol., and Biomedical Engineering: Science and Technol. for Improvement of Health, Safety, and Environ., ICICI-BME 2013*, 2013. 106-110.

[2] Jia, P., Chitta, S. and Manocha, D. 2012. FCL: A General Purpose Library For Collision And Proximity Queries. *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012. 3859-3866.

[3] Ma, Y., Tu, C. and Wang, W. 2012. Distance Computation For Canal Surfaces Using Cone-Sphere Bounding Volumes. *Computer Aided Geometric Design*. 29: 255-264.

[4] Chakraborty, N., Jufeng, P., Akella, S. and Mitchell, J. E. 2008. Proximity Queries Between Convex Objects: An Interior Point Approach for Implicit Surfaces. *Robotics, IEEE Transactions on*. 24: 211-220.

[5] Larsen, E., Gottschalk, S., Lin, M. C., and Manocha, D. 2000. Fast Distance Queries With Rectangular Swept Sphere Volumes. *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, 2000. 4: 3719-3726.

[6] Lin, M. C. and Canny, J. F. 1991. A Fast Algorithm For Incremental Distance Calculation. *Robotics and Automation, 1991. Proceedings, 1991 IEEE International Conference on*, 1991. 2: 1008-1014.

[7] Gilbert, E. G. and Foo, C. P. 1990. Computing The Distance Between General Convex Objects In Three-Dimensional Space. *Robotics and Automation, IEEE Transactions on*. 6: 53-61.

[8] Gilbert, E. G., Johnson, D. W. and Keerthi, S. S. 1988. A Fast Procedure For Computing The Distance Between Complex Objects In Three-Dimensional Space. *Robotics and Automation, IEEE Journal of*. 4: 193-203.

[9] Sulaiman, H. A., Bade, A. and Abdullah, M. H. 2014. Computing Distance Using Internal Axis-Aligned Bounding-Box For Nearly Intersected Objects. In *AIP Conference Proceedings*, 2014. 343-349.

[10] Sulaiman, H. A., Othman, M. A., Ismail, M. M., Meor Said, M. A., Ramlee, A., Misran, M. H. *et al.* 2013. Distance computation Using Axis Aligned Bounding Box (AABB) Parallel Distribution Of Dynamic Origin Point. *2013 Annual International Conference on Emerging Research Areas, AICERA 2013 and 2013 International Conference on Microelectronics, Communications and Renewable Energy, ICMiCR 2013-Proceedings*, 2013.

[11] Jia, P., Sucan, I. A., Chitta, S. and Manocha, D. 2013. Real-time Collision Detection And Distance Computation On Point Cloud Sensor Data. *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013. 3593-3599.

[12] Tang, M., Lee, M. and Kim, Y. J. 2009. Interactive Hausdorff distance Computation For General Polygonal Models. *ACM Trans. Graph*. 28: 1-9.

[13] Quinlan, S. 1994. Efficient Distance Computation Between Non-Convex Objects. *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, 1994. 3324-3329.

[14] Husain, N. A., Rahim, M. S. M., A. R. Khan, Al-Rodhaan, M., Al-Dhelaan, A., Saba, T. 2015. Iterative Adaptive Subdivision Surface Approach To Reduce Memory Consumption In Rendering Process (IteAS) 2015. *Journal of Intelligent and Fuzzy Systems*. 28(1): 337-344.