NEW BOUNDING VOLUME APPROACH FOR DETECTING COLLISION IN COMPUTER ANIMATION

***Abdullah Bade*[1], *Norhaida Suaib*[2], *Abdullah Mohd Zin*[3], *Tengku Mohd Tengku Sembok*[4]**
[1,2] Faculty of Computer Science and Information System,
Universiti Teknologi Malaysia, 81310 Skudai Johor, Malaysia. Email: {abade,haida}@fsksm.utm.my
[3,4]Faculty of Technology and Information Science,
Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia. Email: {amz,tmts}@ftsm.ukm.my

ABSTRACT

This paper presents a method for fast-approximate collision detection between 3D models S undergoing rigid body motion known as oriented convex polyhedra R(S). By enclosing 3D models tightly, the fineness of detected collision can be enhanced. It is known that the large number of void areas which belongs to any 3D bounding volumes B(S) can affect the accuracy of collision detection system. Therefore, a way to compute R(S) using intersection of a set of halfspaces is described. The directions of these halfspaces are generated from calculating covariance matrix. To develop the tightest R(S), the quality of abutting corners by implementing Tribox Bounds method is improved. To detect collision between R(S), a straightforward approach by simply checking its interval pairs in local space system is performed. The proposed approach was implemented and a number of comparisons in terms of time and recorded collision with other B(S) were performed. From the conducted tests, R(S) performs well and might be a possible choice for detecting collision of 3D models undergoing rigid body motion.

Keywords: Bounding Volumes, Collision Detection, Virtual Environments

1.0 INTRODUCTION

The problem of interference checking or collision detection between 3D models in static or dynamic environments is vital in computer animation. In the real world, it is obvious that two solid objects cannot occupy the same space at the same time. In this subject matter, collision detection approaches are certainly needed to preserve the realism of the real world. However, the proposed approaches should be efficient, faster and easy to implement regardless of the number of 3D models that comprise the simulated environment.

Now, denote S as a polyhedron consisting of a finite set of triangulated surface points of the three-dimensional Euclidian Space V. $B(S)$ is convex polyhedra and is used as a geometric bound to enclose 3D model S. The intersection between pairs of objects simply can be confirmed by checking their intersection of $B(S)$. If $B(S)$ intersects, the bound object will have a potential to collide. Otherwise, the bound object will not collide at all.

Most of the previous studies revealed that Convex Hull $P(S)$ might be impractical to be used as a $B(S)$. Some of the reasons are the number of axis test to be considered particularly if one tries to use Separating Axis Test (SAT) and additional time is needed to verify the collision status [2]. Moreover, the construction of P(S) is not straightforward and requires a high memory consumption to represent the P(S) [2]. Though P(S) seems to be unpopular for any practical use, $P(S)$ has the tightest convex polyhedron among $B(S)$ [11], [6].

On the other hand, to utilize the advantage of $P(S)$, the Slab-Based volume $K(S)$ was introduced by [12]. By limiting the number of halfspaces, $K(S)$ is now formed using a set of infinite regions between two parallel planes named as Slab (refer to Fig. 1). To make it work, a number of normal vectors are chosen to determine the direction of the Slab. Meanwhile, by sharing same normal to all 3D sets models, $K(S)$ is now possible to be used as $B(S)$. [13] and [15] explored the idea known as Discrete Orientation Polytopes (k-Dops) and Fixed-direction hull (FDH) respectively. Even though k-Dops or FDH is a form of tight $B(S)$, its major drawback is the updating process. The updating process requires an expensive operation even if the volumes rarely intersect with each other since k-Dops need to maintain the best bound as possible. Several improvements have been cited in the literature and some of them are not straightforward to implement. Some may be easy to implement but is not the optimal solution. The details will be explained in the next section. Therefore, a $B(S)$ approach that has similar shape as k-Dops using modified OBB is proposed. This new $B(S)$ is denoted as Oriented Convex Polyhedra, $R(S)$. Then, several tests are conducted to verify the advantages of $R(S)$ in terms of collision testing, the cost of single collision test and $R(S)$

construction time. The results seem promising especially for 3D models undergoing rigid motion in computer simulated environment.

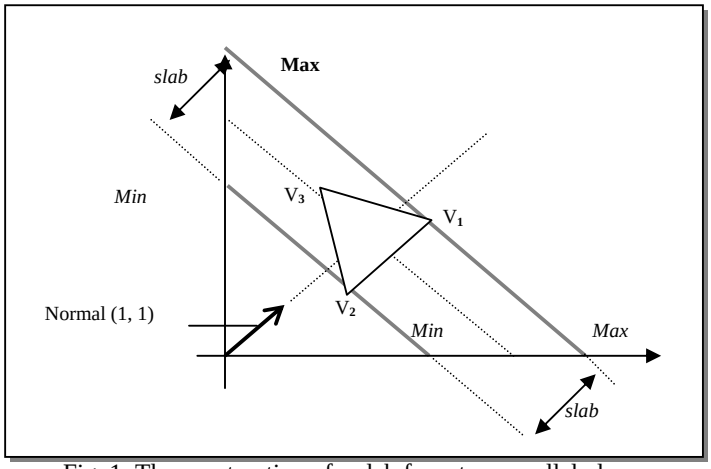


Fig. 1: The construction of a slab from two parallel planes

The remaining sections of this paper are organized as follows: Section 2 explains in brief preliminary research on *B*(*S*). Section 3 touches specific information about the construction of *R*(*S*) using covariance matrix and improvement phase of each abutting corners using [5] approach. Section 4 shows some results of the conducted experiments. The final section summarizes the discussion.

2.0 PREVIOUS WORK

The most common approach in collision detection for n-body system is *B*(*S*). By enveloping complex objects with *B*(*S*), expensive primitive collision checking can be reduced. This is the reason why several researchers manipulate *B*(*S*) as a coarse test before implementing more accurate collision checking [3], [8], [10], [6]. Beside its excellent contribution in collision detection, *B*(*S*) also plays a major role in other fields in computer graphics. Ray Tracing, View Frustum culling and Level of Detail are several examples that manipulate and utilize *B*(*S*) for the sake of performance [19], [17], [1], [18].

The most memory-efficient of *B*(*S*) is Bounding Sphere. The intersection test only manipulates four parameters to verify the existence of collision (one vector for sphere offset and its radian). Furthermore, to do the updating process, it is suggested to utilize the Bounding Sphere offset [4]. AABB is a parallelepiped volume where each of its surface normals coincide with the standard axis system [18]. It can be constructed simply by finding the minimum and maximum coordinate values along each axis. The intersection test of two AABB is performed through comparison of six minimum and maximum coordinate values. Variations of AABB system is clearly explained in [6].

OBB is another efficient *B*(*S*) initiated by [8] and [9]. OBB can be defined as a box where adjacent normal faces are orthogonal. Therefore, fifteen parameters are required to assemble an OBB. These parameters are center point, edge half-length and the orientation vectors that specify three mutually orthogonal unit vectors [8], [9]. Obviously, OBB is a modified AABB specially functions to rotate arbitrarily. Meanwhile, constructing the smallest OBB from a set of point requires a lot of work. Poorly aligned principal axes of OBB will give quite bad OBB fitting. As mentioned in [9] and [2], if the 3D model has the shape of a cube, the constructed OBB sometimes does not fit perfectly. This could be an isolated case due to the equal statistical spread of vertices in all directions. This condition suggests some heuristic to be applied such as using weighted spread of points on convex hull facets [9]. The return is a good fit OBB, but O (n log n) time is required just to compute the convex hull. Another way to handle minimum tight-fitting of OBB is presented by [16].

k-Dops is a convex polytopes. It may be computed by assembling several Slab volumes along k/2 directions [13]. There are several types of k-Dops introduced by [13] known as 14-Dops, 18-Dops and 24-Dops. In general, k-Dops is another adjustment of AABB with several additional directions. As an example, for 18-Dops, three different coordinate systems are required wherein each of them is obtained by rotating the standard axes system by 45° about

one of the principal axis. The most efficient is the 18-Dops [14]. As stated in [14] k-Dops is invariant under translation. Therefore, once 3D model revolves, it will be unaligned with predefined axes. In this condition, one needs to update the volume of k-Dops in order to comply with rotated 3D model. Definitely updating k-Dops operation is expensive. In response to this condition, [13] proposed an alternative approach named as the approximation method. However, the approach constructs larger k-Dops compared to the original volume. Compromising realignment strategy between hill climbing and approximation method as proposed by [7] was not straightforward to implement. Another realignment strategy proposed by [20] based on approximation strategy still produced larger k-Dops. Distinctive Tribox initiated by [5] is not straightforward to implement even though the given technique is able to reduce the number of incorrect abutting corners.

3.0 CONSTRUCTION OF ORIENTED POLYHEDRON, *R*(*S*)

In this section, the construction of *R*(*S*) derived from the intersection of oriented polyhedron is described.

3.1 Preface of *R*(*S*)

Denote *R*(*S*) with nine sets of directions derived from combinations of OBB principal axes namely as u_0, u_1, u_2, $u_0 + u_1$, $u_0 + u_2$, $u_1 + u_2$, $u_0 - u_2$, $u_1 - u_2$ and $u_2 - u_1$ respectively. To form *R*(*S*) bounds, 18 half-spaces are required which are defined by projecting all vertices onto given directions and finding the extreme vertices along each axis. Meanwhile, these 18 half-spaces will produce nine pairs of infinite region of space between two planes known as Slab volumes. Based on a similar idea as reported in [13], *R*(*S*) will form 18-Dop (Discrete orientation polytopes) with the 12 edges been cut off. Nevertheless, the major variation is of course the way to construct the *R*(*S*) orientation in which it manipulates the principal axes of OBB. Fig. 2 shows the generated *R*(*S*) that bounds two sample 3D models. The model given by Stanford Computer Graphics Lab and Cyberware, Inc. was used in this testing.

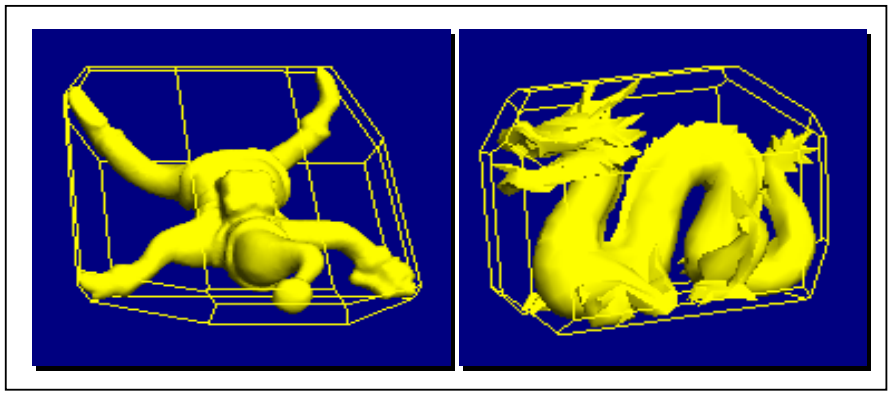


Fig. 2: Two sample 3D set models (Santa and Dragon) bounded by *R*(*S*)

3.2 *R*(*S*) Representation

The construction of *R*(*S*) is quite straightforward. Several calculations using Principal Component Analysis algorithm (PCA) were performed to construct the orientation of *R*(*S*). In particular, the mean and covariance matrix of triangulated surface points were computed. This is the major type of 3D model representation in this instance.

Now, denote x^i, y^i and z^i correspond to the triangulated surface points, μ, C_{ij}, n represents mean, 3 by 3 covariance matrix and number of triangles respectively. The following equations explain the above process (see equation 1 and 2). The rest which follows are explained in the following steps (see step 3-7).

1. Let $\mu = \frac{1}{3n} \sum_{i=0}^{n} p_i + q_i + r_i$
2. Let $C_{ij} = \frac{1}{3n} \sum_{i=0}^{n} (p_i p_j + q_i q_j + r_i r_j)$
3. Replace each p, q and r with p-μ, q-μ, and r-μ, respectively.

4. Compute the eigenvectors of symmetric matrix C_{ij} and normalize the value before using them as a principal axes. The eigenvectors will likely be aligned to the principal axis with the geometry of object.

5. Then, to construct the directions of $R(S)$, three other different coordinate systems are required wherein each of them is obtained by rotating the original coordinate system of OBB (gathered in the previous step - the eigenvectors of symmetric matrix) by 45° about one of the principal axis. In this case, in order to construct these three coordinate systems, only the original principal axis (orientation) generated by eigenvector function were manipulated. In other words, the halfedge-lengths and center point for each of the new coordinate system as needed by OBB representation were not used and computed. The major reason is, by sharing similar center point and halfedge-length for each of them, the unfit volume of $R(S)$ will be formed. Fig. 3 shows the situation clearly. The new coordinate system after 45° rotation about one of the principal axis shows that, if it shares similar halfedge-length and center point, the unfit blue box will be produced. Meanwhile, if only the generated direction after rotating 45° is used, the fit red box will be generated. In this case, the red box has its own halfedge-length and center point. Since calculating the half-length and the center point of each generated coordinate system is not done, a number of computations can be reduced slightly. In Fig. 3, the 2D volume shows the original OBB in green while the blue box is the generated volume using new coordinate system that shares identical halfedge-length and center point with original OBB. The red box shows an optional volume that manipulates the direction after being rotated 45° without sharing identical halfedge-length and center point. The 3D volume shows similar situation wherein one of the coordinate system will produce fit volume (original) while the others produce unbalance volume.

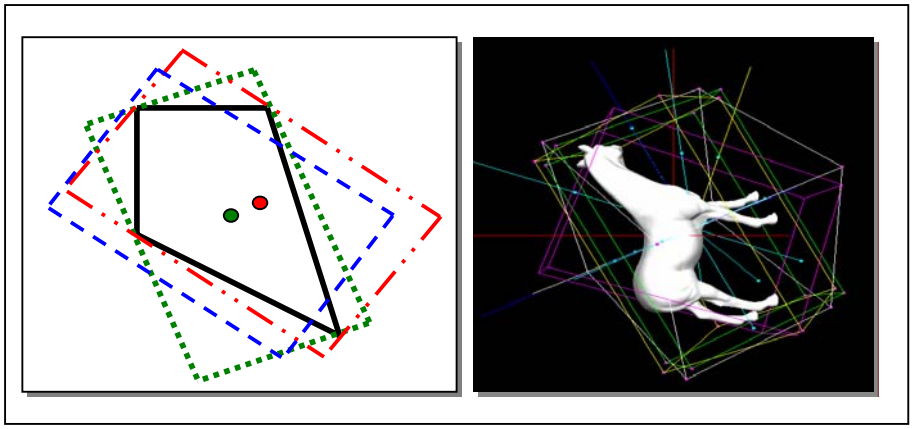


Fig. 3: The 2D volume and the 3D volume
(Please download the softcopy to view the colors.)

6. In order to construct a bounding box, project all vertices onto the principal axes and find the extreme vertices along each axis.

7. After that, to improve the quality of abutting corners, the technique proposed [5] is applied. In this case, improving the quality of abutting corners is important since the generated intersection points will be used for intersection testing. First, there is a need to find out the intersection of three abutting planes. This is followed by finding any changes of the local topology at a corner based on given condition as cited in [5]. In general, there are four type of topological configuration at a corner. Given the 18 projection bounds, the topology of the boundary at each corner may be easily characterized by calculating the intersection point p and testing it against the three principal planes (refer to Fig. 4). As mentioned by [5], each corner should possess three main planes. Denote each of them as h_A, h_B and h_C respectively. Associate chamfer planes are denoted as h_{AB}, h_{AC} and h_{BC}. The best explanation about the way to obtained intersection points can be found in [5].

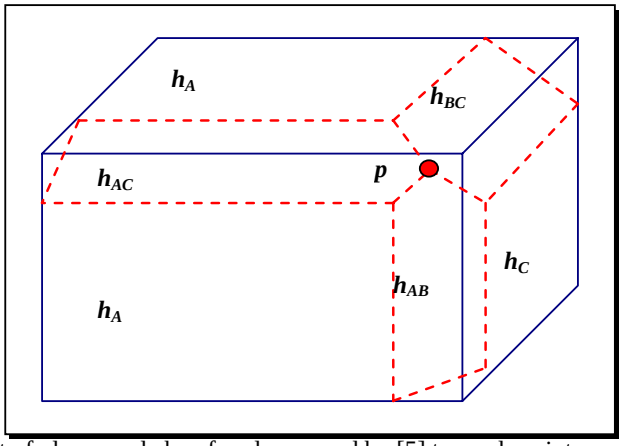


Fig. 4: Set of planes and chamfer planes used by [5] to produce intersection point *p*

4.0 INTERSECTION TESTING

To perform an overlap test between two $R(S)$ volumes, it is a must to transform both of them into a common coordinate system. In this case, one of the $R(S)$ volumes was transformed into the local coordinate system of the other volume. This step saves a lot of the computations rather than performing a similar transformation to both of the volumes into a world coordinate system. The intersection testing concept is derived from a similar concept as k-Dops intersection test. The testing is easy to implement and performs a fairly faster collision checking. Although the $R(S)$ volume is constructed by manipulating similar orientation of OBB, the same collision detection approach known as Separating Axis Test (SAT) to detect collision cannot be simply applied. This is so for orientations higher than three, $k > 3$ where k represents the number of orientation, number of axis test to be considered increases drastically. As mentioned in [2], if the number of edge directions is at most $3k$-6, therefore, by using Euler's formula, the total number of axes to be considered are $2k + (3k\text{-}6)^2$. Therefore, the following algorithm explains the collision checking between two $R(S)$ volumes. In general, it starts with transforming one of the $R(S)$ volumes into the other volume coordinate system. This is followed by comparing all intervals of each volume to testify the existence of overlapped region.

```
Testing Algorithm (Object_A , Object_B)
{
    Pre: Subject contains geometry information of R(S)
         that is Intersection Point P.
         Current Transforming Matrix of each subject.

    Post: Both objects overlapped.

    Return: True if all intervals (SLAB) are
            overlapping, false if any of the intervals
            do not overlap.

    1   Compute Transformation Matrix expressing
        Object_B into Object_A 's coordinate frame.
    2   Project all 32 corners of Object_B  into
        Object_A 's directions.
    3   Loop (i=0;i<9;i++)
           If(Object_A.min[i] > Object_B.max[i] ||
                    Object_A.max[i] < Object_B.min[i])
           return 0; // no overlapping detected.
    4   return 1; // all interval are overlapping
                  // both R(S) have a collision.
}
```

5.0 IMPLEMENTATION AND RESULTS

To evaluate the approach, all experiments given in this section have been conducted on a Pentium IV PC with 2.8 GHz, 1 GB main memory and NVIDIA GeForce4 MX 440 with 8X AGP. Two types of experiments were conducted. The first experiment measures the number of detected collision for n objects within a particular time frame. The second experiment quantifies the pre-processing time to generate *R*(*S*) and time to detect a single collision.

Number of Detected Collision :– In this experiment, a number of collisions of 3D model undergoing rigid body motion were verified. A variety of 3D models were used in a simulated environment. Most of the models were provided by Stanford Computer Graphics Lab. Examples of 3D models used are Santa, House, Cow, Skull, Bunny, Dragon and Dinosaur1. The first experiment is conducted by placing 20 varieties of 3D model randomly in space and testing them for intersection (Fig. 5). The space or the simulated environment is a cube that consists of Santa, Horse, Cow, Dragon, Bunny and K-Dops models (two models each). In addition, the space also has three Ellipses and one model each of Venus, Diamond, Hand, Skull and Sphere. Each of the 3D models has arbitrary orientation.

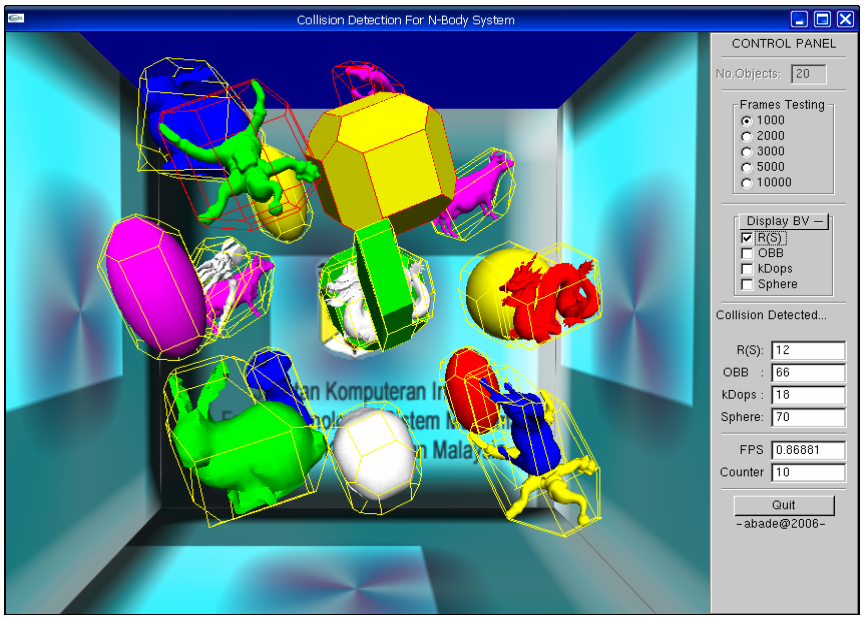


Fig. 5: A simulated environment comprises of various 3D models

In this experiment, the approach was benchmarked in terms of the number of detected collision(s) with OBB and k-Dops since the approach is derived from those concepts. Bounding Sphere is the worse case key indicator. From the results of the experiment conducted, it is found that, *R*(*S*) performs fairly favorably compared to bounding Sphere, OBB and k-Dops (Fig. 6). On average, the number of collisions detected by *R*(*S*) is 38.34% less than OBB. In the meantime, the number of collision detected for every time frame for k-Dops is an average of 1.1037% more compared to *R*(*S*). Using similar experiment properties, several experiments were conducted by manipulating 20 identical 3D models.

[1] Thank you Stanford Computer Graphics Lab (http://graphics.stanford.edu/data/3Dscanrep), Cyberware Sample Models (http://www.cyberware.com) and GIT Large Geometry Models Archive (http://www.cc.gatech.edu/projects/large_models/)

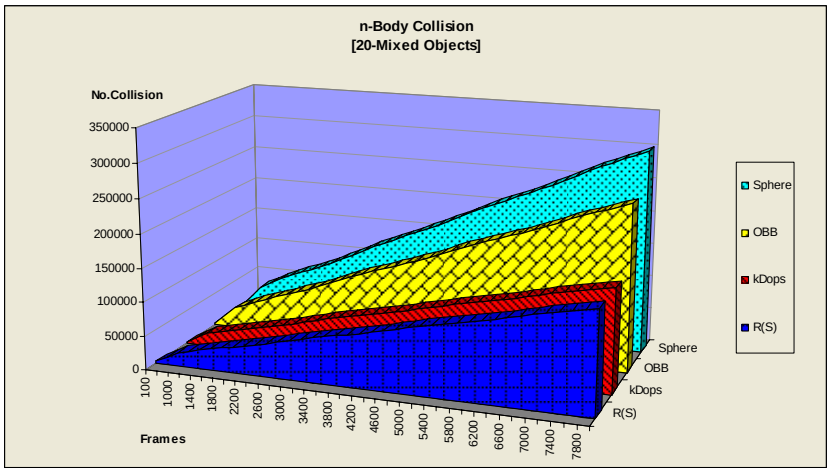


Fig. 6: Collision detection for n-Body Objects (20- Mixed 3D models)

The first experiment was executed to test the intersection of 20 Cow models. In this experiment, once again it was found that $R(S)$ yielded better performance when compared to other $B(S)$. Average performance of $R(S)$ in terms of detected collision is 26.625% and 18.979% less compared to OBB and k-Dops approach respectively (Fig. 7).

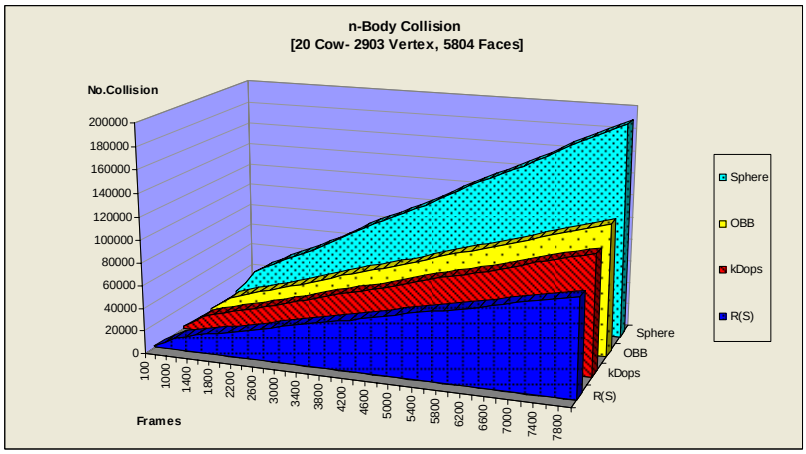


Fig. 7: Number of collision detection for 20 Cows recorded frames ranges 100 to 8000 frames

The next experiment was to validate the number of collisions recorded between 20 Ellipses. Using similar experiment properties as 20-Mixed models, it was found that, for such lozenge 3D models, $R(S)$ performed comparatively better than OBB and k-Dops (Fig. 8). As an example, the number of collisions recorded for OBB specifically in frame 3200 and 3400 are 25.0698% and 24.999% more when compared to $R(S)$. Meanwhile, using similar time frames, the extra numbers of collisions are 3.4636 % and 3.401% respectively as traced by k-Dops compared to $R(S)$.

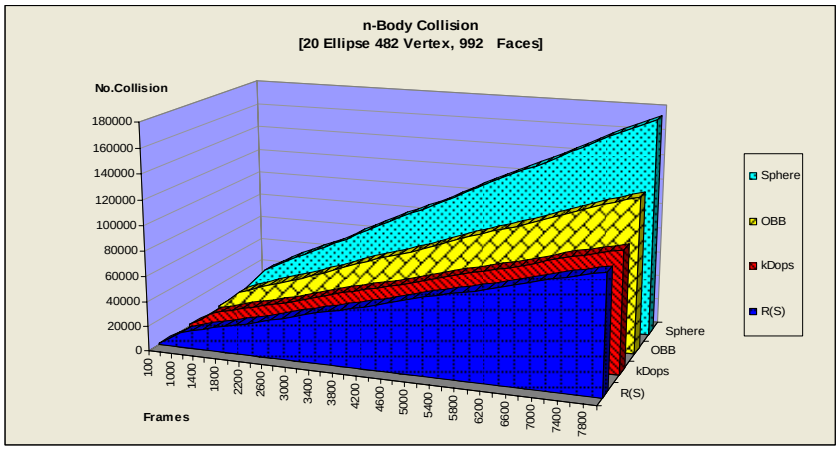


Fig. 8: Number of collisions recorded for 20 Ellipse models in the simulated environment

A few more experiments were carried out in order to analyze the number of recorded collision for various identical models. For example, the experiment between 20 Horses (Appendix A), 20 Santas (Appendix B), 20 Dragons (Appendix C), 20 Hands (Appendix D), 20 K-Dops models (Appendix E) and 20 Diamonds (Appendix F). Throughout the experiments, it was found that, *R*(*S*) works well especially for irregular models that have gradual outliers such as the Santa, Dragon, Horse, Cow, Dragon and Hand. This is due to its capabilities in reducing a large number of empty corners by cutting off 12 edges of the OBB volume and no tumbling operation is needed in order to comply with rotated bounded models. On the other hand, for objects such as the Diamonds and K-Dops models, *R*(*S*) successfully produced tight-fitting volumes and contributed superior performances almost in every time frame.

An experiment to determine the most accurate B(S) by manipulating one of them as an object was also performed. To setup that situation, spheres were selected to be the experiment subject (Fig. 9). By doing so, it gives an advantage to the bounding Sphere and gives the most accurate collision detection. Therefore, 20 sphere models were used where each of them has 467 vertices and 992 faces. Using similar experiment properties as the previous, R(S) is worse compared to k-Dops (Appendix G). Average performance showed that 14.654% more collision was detected by R(S) as compared to k-Dops. Models which have equal statistical spread of vertices in all directions will sometimes produce worse R(S) bounds. Moreover, the sphere model itself easily aligns well with the axes of the k-Dops and gives a fair advantage to k-Dops. From the experiment, the number of collision detected by R(S) is about 47.2258% less than OBB.

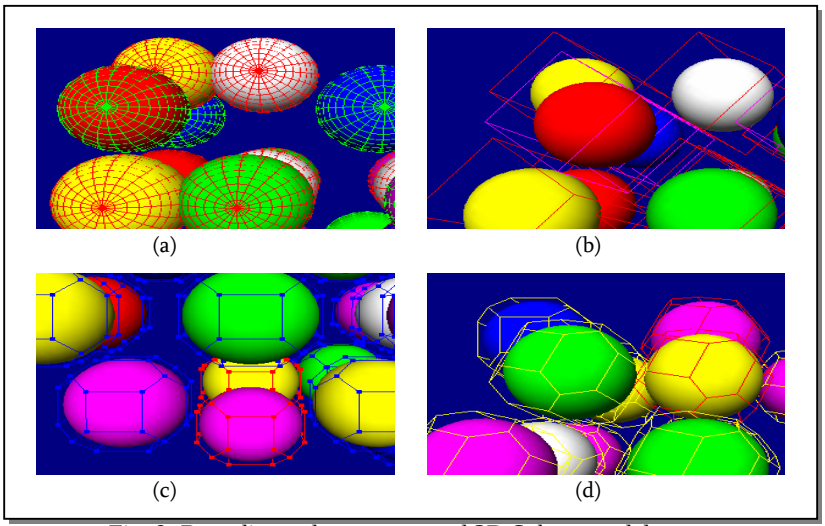


Fig. 9: Bounding volumes surround 3D Sphere models:
(a) using bounding Sphere (b) using OBB (c) k-Dops and (d) *R*(*S*)

Preprocessing Time and Time to Detect Single Collision :– Fig. 10 shows the time to detect a single collision for *R*(*S*) using the algorithm explained in section 4. In this experiment, 1000 collision checking between two *B*(*S*) were tested and the time it takes to certify a single collision was recorded. From the experiment, it was found that by using the algorithm explained in section 4, *R*(*S*) is able to confirm collisions much faster compared to OBB and k-Dops. Although the min-max values of 32 corners of *R*(*S*) were compared, the performance does not drop off much. k-Dops obviously takes more time since it needs to update and realign to the bounded model. The SAT used by OBB to verify collision is fairly good and fast. The best *B*(*S*) is the bounding Sphere since it only needs to compare the sphere radii to validate collision.

The latest experiment is to compute the preprocessing time to generate single *R*(*S*) of several 3D models (Fig. 11). The preprocessing time needed by *R*(*S*) is the highest compared to other *B*(*S*). Therefore it can be deduced that one of the major causes that increase the preprocessing time is the computation and the procedure to improve the quality of abutting corners. Finding the intersection point of abutting corners is important for *R*(*S*) since it will determine the accuracy of *R*(*S*) volume. Fig. 11 shows that preprocessing time needed by *R*(*S*) to calculate the enclosing volume is in the average of 16.3% higher than k-Dops. Once again, finding the quality of abutting corners contributes to the significant increment of time. On the other hand, comparing *R*(*S*) to OBB might be inappropriate since *R*(*S*) manipulates 9 orientations to develop *R*(*S*) volume while OBB only needs 3 orientations to generate an orientated box. In this case, of course *B*(*S*) such as k-Dops and *R*(*S*) will need extra preprocessing time since more orientations should be considered.

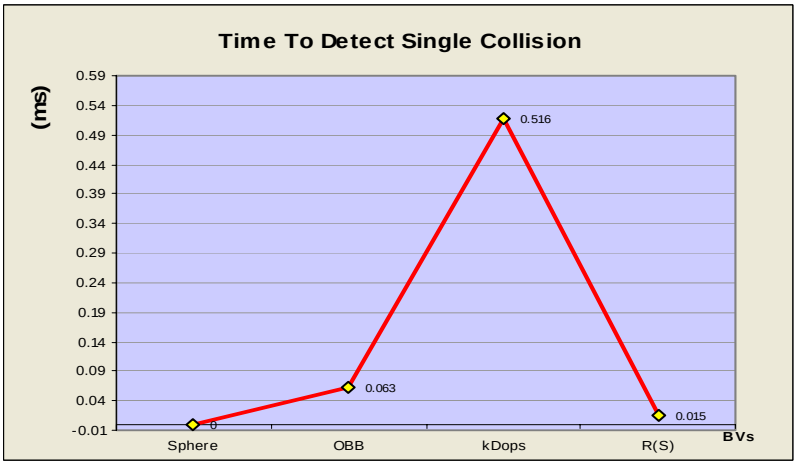


Fig. 10: Time to detect single collision of *B*(*S*)

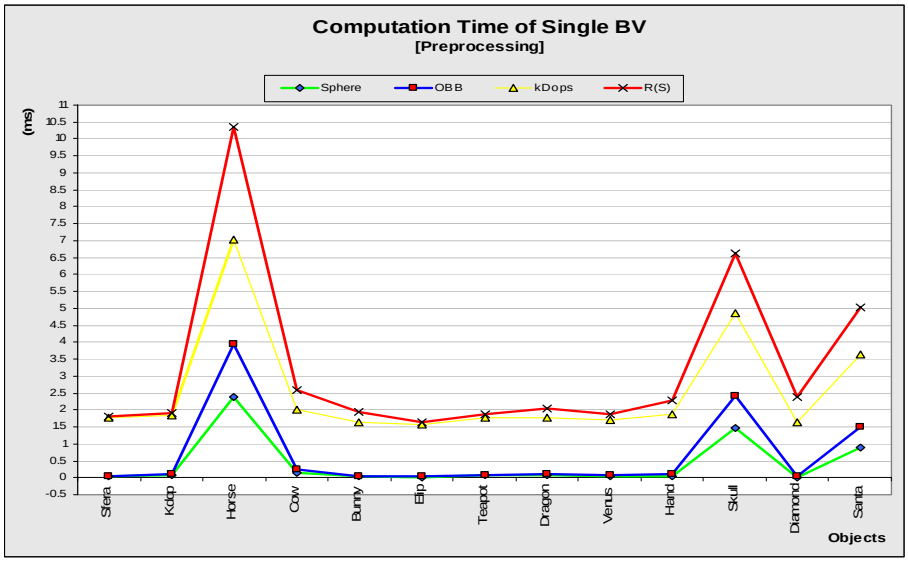


Fig. 11: A preprocessing time needs to compute *B*(*S*) for several 3D models

6.0 CONCLUSION

In this paper, a method for detecting collision between 3D models was presented. The approach is made possible by computing an $R(S)$ using intersection of a set of halfspaces. The directions of these halfspaces are formed from the calculation of covariance matrix. The collision detection scheme that was implemented shows fairly good results in terms of collision checking and time to detect single collisions. Currently, several techniques in order to reduce the preprocessing time needed to construct $R(S)$ are being investigated. One of the potential techniques is by using approximate convex decomposition and manipulating simple $B(S)$ before calculating the abutting corners procedure.

ACKNOWLEDGEMENT

Special thanks to Mohamed Fauzi Othman (Universiti Teknologi Malaysia), Assistant Professor Rafael Cabredo (De La Salle University, Manila) and Syaiful Nizam Yahya (VE Lab, GMM) for their supportive ideas and valuable critiques in terms of writing and programming.

REFERENCES

[1] U. Assarsson, And T. Möller, "Optimized View Frustum Culling Algorithms for Bounding Boxes". *Journal of Graphics Tools***,** Vol. 5 Issue 1, 2000, pp.9 – 22.

[2] G.V.D. Bergen, *Collision Detection In Interactive 3D Environments*, Morgan Kaufmann Publishers, USA, 2003.

[3] G.V.D. Bergen, "Efficient Collision Detection Of Complex Deformable Using AABB Trees". *Journal of Graphics Tools*, Vol. 2 No. 4, 1998, pp. 1-13.

[4] F.Chin-Shyurng, And. W. Jui-Lung, "Efficient Time-Interrupted and Time-Continuous Collision Detection Among Polyhedral Objects in Arbitrary Motion". *Journal Of Information Science And Engineering*, 15, 1999, pp. 769-799.

[5] A.Crosnier, And J.Rossignac, "Tribox bounds for three-dimensional objects". *Computers & Graphics*, Vol. 23 No.3, pp. 429-437.

[6] C. Ericson, *Real Time Collision Detection*, Morgan Kaufmann Publishers, USA, 2004.

[7] C. Funfzig, And. D.Fellner, "Easy realignment of k-Dop Bounding Volumes". *Proceeding of Graphics Interface 2003*, Nova Scotia, Canada, pp. 257-264.

[8] S.Gottschalk et al., "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection". *In Computer Graphics Proceedings*, Annual Conference Series, ACM SIGGRAPH. New Orleans, Louisiana, pp. 171-180.

[9] S.Gottschalk, 2000. "*Collision Queries Using Oriented Bounding Box*", Ph.D Thesis, Department of Computer Science, University of North Carolina, Chapel Hill.

[10] P.M. Hubbard, "Collision Detection For Interactive Graphics Application'. *IEEE Transactions on Visualization and Computer Graphics*, Vol.1 No.3, 1995, pp. 218-230.

[11] D.Halperin, And M.Sharir, "New bounds for lower envelopes in three dimensions with application to visibility in terrains", *Proceedings of 9th ACM Symposium on Computational Geometry*. 1993, pp. 11-18.

[12] T.L. Kay, And J.T. Kajiya. "Ray Tracing Complex Scenes", *Computer Graphics (SIGGRAPH '86 Proceedings*. Vol. 20, 1986, pp. 269-278.

[13] J.T. Klosowski, 1998. "*Efficient Collision Detection For Interactive 3D Graphics And Virtual Environment*", Ph.D Thesis, State University of New York, Stony Brook.

[14] J.T. Klosowski et al., "Efficient Collision Detection Using Bounding Volume Hierarchies Of k-Dops". *IEEE Transaction on Visualization and Computer Graphics*. Vol. 4 No.1, 1998.

[15] Petr. Konecny, 1998. "*Bounding Volumes in Computer Graphics*". MSc Thesis, Faculty of Informatics, Masaryk Universitiy, Brno, Czech Republic.

[16] M. Lahanas et al., "Optimized Bounding Boxes for Three-Dimensional Treatment Planning in Brachytherapy". *Medical Physics*, Vol. 27 No.10, 2000, pp. 2333-2342.

[17] T.Larsson, And T.Akenine-Möller, "Strategies for Bounding Volume Hierarchy Updates for Ray Tracing of Deformable Models". MRTC Report, 2003.

[18] T.Moller, And E.Haines,. *Real-Time Rendering*. A K Peters Ltd, 1999.

[19] G.Müller et al., "A Rapid Clustering Algorithm for Efficient Rendering". *Eurographics Association*, 1999, ISSN 1017-4656.

[20] G.Zachmann, "Rapid collision detection by dynamically aligned DOP-Trees". *Proceeding of IEEE Virtual Reality Annual International Symposium* (VRAIS 1998), Atlanta, Georgia, 1998, pp. 90-97.

BIOGRAPHY

Abdullah Bade obtained his Master of Computer Science (research) from Universiti Teknologi Malaysia in 2002. Currently, he is a lecturer at the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia. His research areas include computer graphics application in particular collision detection area, bounding volumes and physical based modeling. He has published a number of papers related to these areas. He is also an affiliate member of ACM SIGGRAPH since 2005.

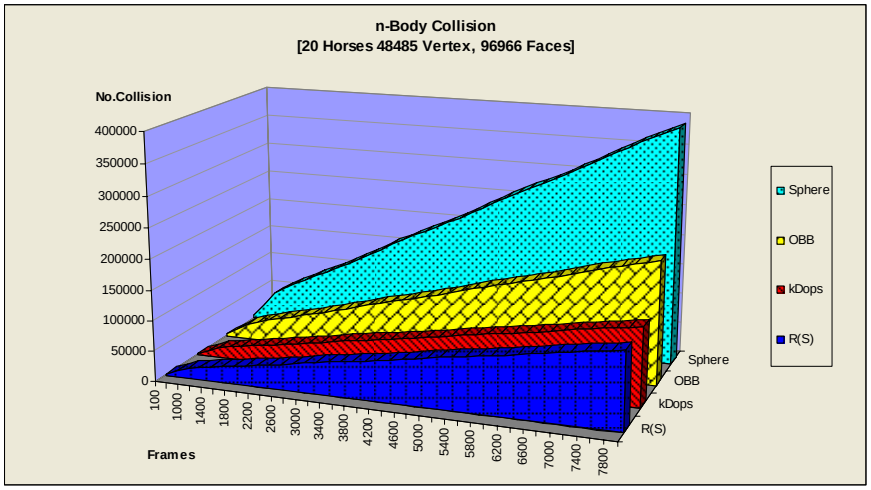
Norhaida Mohd Suaib obtained her MSc. in Computer Graphics and Virtual Environments from University of Hull, United Kingdom (1997). Besides lecturing at the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia, she is also an active researcher. Her research interests include computer graphics application and virtual reality. She has published a number of papers related to these areas. She is also an affiliate and active member of ACM SIGGRAPH, as well as the KL-SIGGRAPH since 2003.

Abdullah Mohd Zin is a full professor at the Department of Industrial Computing, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia. He obtained his PhD (Computer Science) from Nottingham University (UK). He has published numerous articles in international peer-reviewed journals. His research interest includes collaborative learning tools, software management, E-Learning and foundation of programming. Currently he is the deputy dean as well as the head of the programming research group at the faculty.

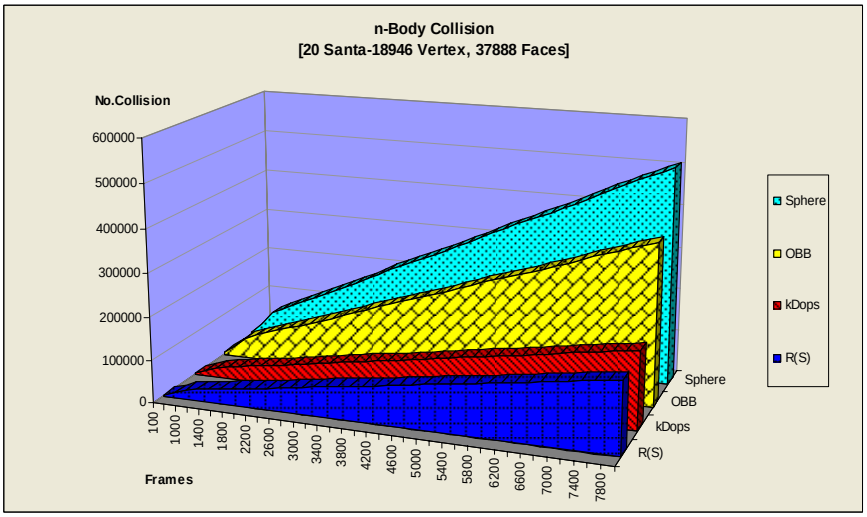
Professor Dr. Tengku Mohd Tengku Sembok has over twenty years experience in various fields of Information Technology and has managed numerous R&D and consultancy projects successfully. He obtained his PhD from Glasgow University in 1989. He is currently holding the chair of professor in Computer Science in Universiti Kebangsaan Malaysia and has held several academic posts, among them are Head of Computer Science Department and the Dean of Faculty of Information Science and Technology. He sits in various national committees on IT such as Terengganu State Steering Committee on IT, IRPA Steering Committee in Services Sector (MOSTE), and Curriculum Development Committee for Computing Subject (Malaysia Examination Council). Professor Tengku Mohd is involved in the MSC Smart School Flagship Application as Project Director in the development of Mathematics courseware for secondary schools. His current research areas are intelligent information system, information retrieval, multimedia courseware, and knowledge management.

Appendix

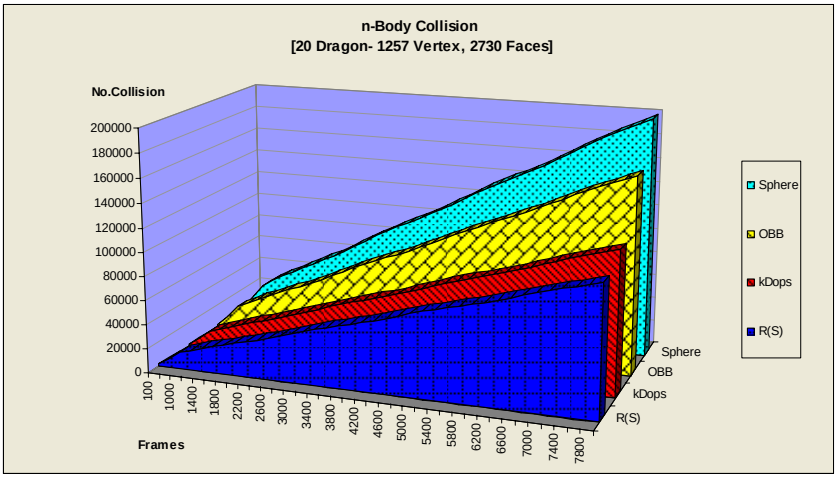
A: Collision detection between 20 Horse models.



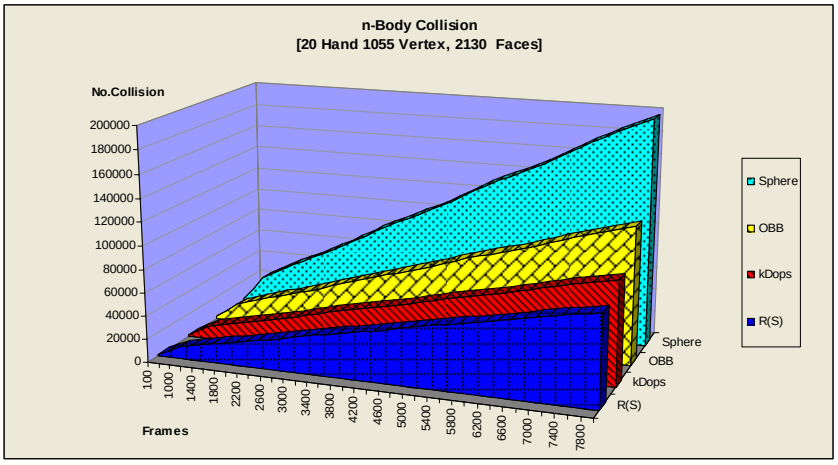
***B*:** Collision detection between 20 Santa models.



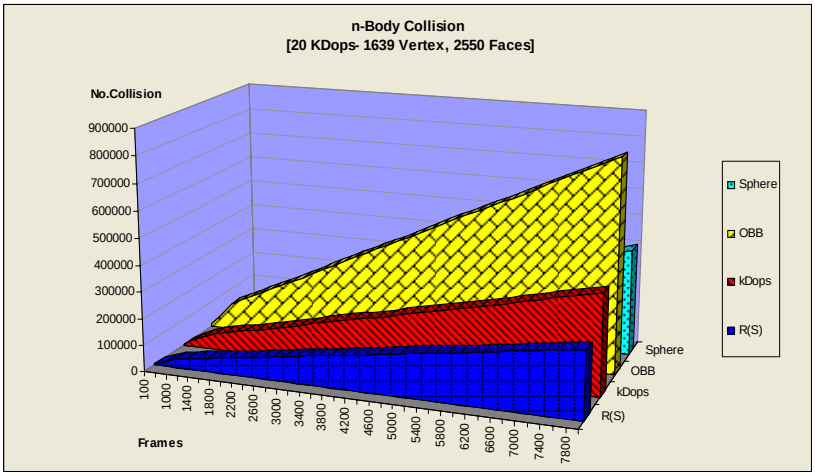
C: Collision detection between 20 Dragon models.



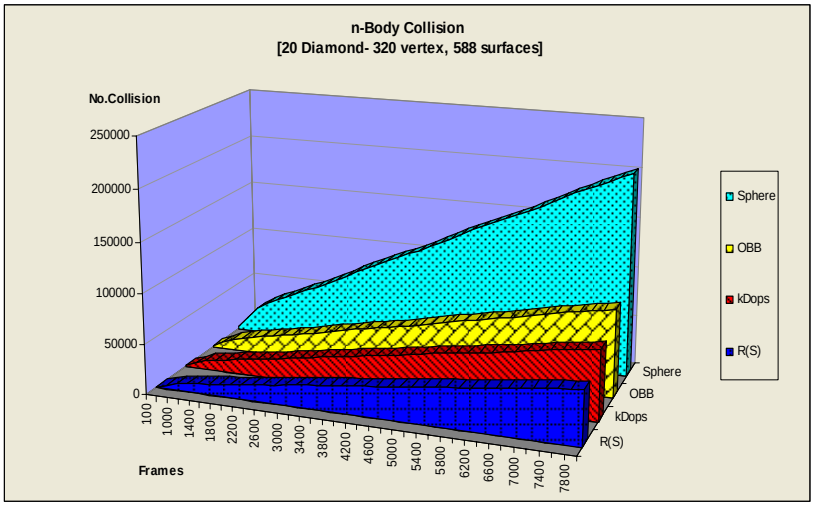
D: Collision detection between 20 Hand models.



E: Collision detection between 20 Kdop models.



F: Collision detection between 20 Diamond models.



G: Collision detection between 20 Sphere models