EUROPEAN ACADEMIC RESEARCH
Vol. I, Issue 10/ January 2014

ISSN 2286-4822
www.euacademic.org

Impact Factor: 0.485 (GIF)
DRJI Value: 5.9 (B+)

Gilbert-Johnson-Keerthi Algorithm for Computing the Shortest Distance between Two 2D Convex Hull Polygons Based on Andrew's Monotone Chain Hull Algorithm

ASIA MAHDI NASER ALZUBAIDI
Computer Science Department, College of Science
Karbala University
Karbala, Iraq
MAIS SAAD AL-SAOUD
Computer Science Department, College of Science
Karbala University
Karbala, Iraq

Abstract:

This paper provides an overview of the Gilbert, Johnson and Keerthi algorithm (GJK) for measuring the minimum distance from one convex object to another in linear time complexity. Also, GJK regards as an effective and reliable method for obtaining the couple of closest points between the shapes. Measuring the shortest distance is ubiquitous in diverse set of applications such as robotics, computer graphics, real-time animation, virtual environment Simulation and Collision Detection. In this paper we put forward a GJK technique in order to solve the minimum distance problem between geometric bodies. One of the more powerful features of GJK is that it can only work on convex shapes, so we initially generate the two sets of 2D points and then apply Andrew's Monotone Chain algorithm to obtain the convex boundary of the two shapes. Finally we use GJK algorithm to determine the shortest distance. Actually, instead of iteratively finding of the closest distance between the features of two convex shapes, GJK algorithm uses a specific mathematical concept called the Minkowski Sum/Difference and then building the simplex polygon inside it by using support mapping function and finding the closest difference between their Minkowski Difference and the origin point in 2D space. In order to increase the robustness of GJK algorithm we

modify the terminating conditions of it. The experimental results shows that the time execution of GJK algorithm is fixed and independent of the geometric difficulty of the convex objects.

Key words: GJK distance algorithm, Closest features, Simplex, Convex hull algorithms, Support function and farthest point, Minkowski Difference.

1. Introduction

Many of algorithms for calculating the Euclidean distance between objects in m-dimension have been developed and adopted for use in many areas such as Virtual Reality systems, robot motion planning and physical simulators. In particular calculating the minimum Euclidean distance between geometric shapes has received much attention to avoid the bottlenecks and also because it serves as a fundamental construct in many collision detection packages with convex bodies (Senin et al. 2003). In this paper we give an insight for distance computations based on GJK distance algorithm in linear time complexity, dependent on the number of vertices for each object. Moreover, it is not constrained to a specific number of dimensions and therefore can be used in any m-dimensional space (Lindemann 2009). Although the GJK algorithm is difficult to grasp at first, it is the popular and most famous algorithm in computing the shortest distance between sets of convex objects. The main idea of algorithm is to iteratively search about the closest point in the subset of generated simplex inside the Minkowski Difference to the origin point (0,0) and eliminate the nonessential vertices to determine the nearest so the shortest distance is the magnitude of the nearest point (Wang et al. 2012). The GJK algorithm constructs simplex by reoccurrence of support mapping function which is the inner products of a specific direction vector with all points of convex shape and searching for the largest output to determine the

farthest point. Then, the support function is the difference between the farthest point for each object. Finally, add it to simplex vertices if it the nearest point on line segment to origin point in that direction (Liu et al. 2001). This introduction gave a brief outline over the GJK algorithm and the rest of this paper is arranged as following: After a discussion of related backgrounds and provides some information about the algorithm's history in Section 2, Section 3 discusses the Andrew's Monotone Chain convex hull algorithm. While in section 4 we review the basic mathematical background and knowledge that needed to deeply grasp the functionality of GJK for our research, which involves the demanding method of computing the support mapping function of set of points and the formulation of the closest distance problem with explaining of how it can be resolved by calculating the closest point on line segment function. The actual GJK algorithm will be presented in Section 5. Section 6 gives the experimental results of GJK algorithm with test examples and section 7 provides the paper's conclusions.

2. Related Work

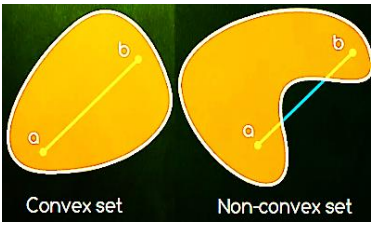
The shortest Distance between pair of body problem has been widely studied in the previous works of computational geometry, robotics and computer graphics. Various techniques and popular algorithms have been proposed and implemented with the purpose of speeding up the computing distance between convex bodies. Actually the origin of GJK algorithm was described by E.G. Gilbert, D.W. Johnson and S.S. Keerthi in 1988 restricted to computer the distance between pair of convex polyhedral objects without any enhancement and preprocessing, it takes linear time depending on the number of polyhedral shape vertices and the support function that used to describe it (Gilbert et al. 1988). Both E.G. Gilbert and C.-P. Foo In 1990 introduced an enhancement of original GJK algorithm

to manipulate all types of geometric convex shapes (Gilbert and Foo 1990), while, in 2001 some researchers introduced paper involving an algorithm to determine the shortest Euclidean distance made by GJK between convex polyhedral object represented by the convex hull of its points in 2D and 3D space. They found that the algorithmic improvement in distance computation algorithm giving simple and proficient algorithm for depicting out the information of where the closest point on a convex polyhedron object to a origin point is (Liu et al. 2001). In 2009, P. Lindemann published a paper that gives an overview of the enhanced GJK method for finding the Euclidian distance between two convex sets and used Voronoi regions for differencing whither the closest point to origin is edge or vertex in convex shape and concluded that the GJK algorithm adapted to solve the actual problem and parts of it replaced by more suitable procedures to improve time complexities near to constant time (Lindemann 2009).

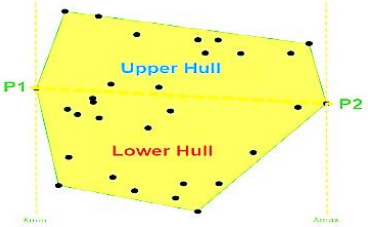
3. Convex Hull Via Andrew's Monotone Chain Algorithm

Constructing a convex hull is one of the first important geometry algorithms. There are various computing convex algorithms and the most common form of this geometric algorithms involves determining the convex hull of set of m-dimension points in plane using bounding boxes techniques (Sacristan 2012). Mathematically, set S of points $P \in R^m$ is named convex and bounded if for any pair of vertices a and b such $a,b \in P$ then, ab line segment lies entirely in S, as shown in figure (1). In R^2 convex hull for both Graham scan and Andrew's Monotone Chain scan have similar in idea and to implement them we need to use stack structure. But in practice, Andrew's algorithm will execute slightly faster (GeomAlgorithms 2012). Andrew's Monotone Chain convex hull algorithm is an algorithm which creates convex hull of a set of 2D points in O (nlogn) time duo to the needing of firstly sort the

points lexicographically by their x-coordinate and then constructing upper hull from its rightmost point p_2 to the leftmost point p_1 in counter clockwise order and create the lower hulls of the points from leftmost point to rightmost point in the same orientation with O(n) time(Sacristan 2012). As depicted in figure (2).



Figure(1) Convex and Non Convex hull



Figure(2) Andrews Monotone Chain Algorithm

To build the upper and lower convex hull we need to iteratively test wither the points according to p1p2 line is above, on or under it by fast accurate computation of cross Direction routine

Cross_Direction algorithm (GeomAlgorithms 2012).
Input: three points P0, P1, and P2 .
Output: value
> 0 for P2 above of the line through P0 and P1
= 0 for P2 on the line through P0 and P1
< 0 for P2 under of the line through P0 and P1
begin
step1:

$$\text{value} = (P1.x - P0.x) \times (P2.y - P0.y) - (P2.x - P0.x) \times (P1.y - P0.y)$$

step2: return value
finish
figure(3) show set of 2D points. P8, P10 are the leftmost and rightmost points respectively. P1, P3 for upper hull. P9, P7 for lower hull.

Figure (3) set of points and their convex

Andrews Monotone Chain Algorithm(Sacristan 2012),(GeomAlgorithms 2012)(PAVZAV 2010).

Input: set $P = \{p = (p.x, p.y)\}$ of n points.

Output: set H = the output convex hull of set P, k number of output convex points
in set H.

Begin

Step1: Sort points set P based on their X coordinate in counter-clockwise order,
make P as the sorted array of N points.

Step2: Let P1 is leftmost point, P2 is the rightmost point, k=3.

Step3: Add both points p1,p2 to output array of convex points H as H[1]=p1,
H[2]=p2.

Step4: Divides the set of points into upper and lower hull based on the line p1p2.

Step5: Build upper hull as follow

- for I = 1 to n-1
- begin
- while (k >= 3) and (cross_Direction (H[k-2], H[k-1], p[I])<= 0) do
- k = k - 1
- H[k] = P[I]
- k= k + 1
- End

Step6: Build lower hull as follow:

- t= k + 1
- for I:=n-2 down to 1 do
- begin
- while (k >= t)and(cross_Direction(H[k-2], H[k-1], P[I]) <= 0) do
- k =k - 1
- H[k] = P [I]
- k = k + 1
- End

Step7: return H and k.
Finish

4. Mathematical Preliminaries

Now we review the mathematical background that is vital to understand the functionality of GJK algorithm, such as simplex, support function and farthest point, closest points on line segment and Minkowski difference.

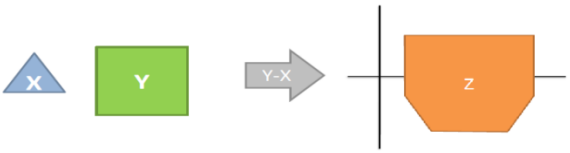
4.1. Minkowski Difference of Two Polygons

GJK algorithm uses the fact that the Euclidian distance between two convex polygons is the shortest distance between their Minkowski Difference and the origin. This function can be used in many applications, such as motion planning and computer-aided design and manufacturing. Computing the Minkowski routine of Two Polygons is by taking the Minkowski sum of object and the mirror of another object. So, the second shape gets flipped about the origin and all of its points are negated (CGAL 2013) (Bernabeu et al. 2013), as illustrated in equation(1).

$$Z = \{ Yj - Xi : Xi \in X , Yj \in Y \} \qquad \dots (1)$$

Actually computing Minkowski operation is not very practical as the number of points needed is |X|*|Y| = O(n^2) if we

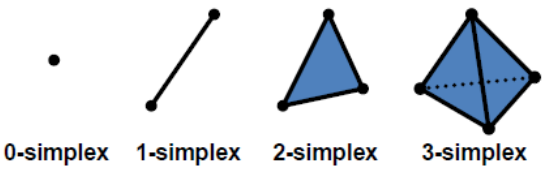
assume that the number of vertices in both shapes = n become large. Therefore, instead of computing the Minkowski difference set points we will construct it implicitly through using a support mapping function to build a polygon inside the Minkowski Difference, called the simplex . Then we compute the distance between simplex and origin point (Twisted Oak Studios 2012). Figure (4) shows two polygons and their Minkowski difference.



Figure(4) Minkowski difference between 2D polygons

4.2. Simplex

In geometry, a simplex can be defined as the convex polygon of a set of (m+1) vertices in some Euclidean space of dimension m or higher. So a simplex is a generalization of shape such as line, triangle, tetrahedron to arbitrary dimensions. In mathematics, a simplex is denoted as the smallest convex set containing the given vertices. For example, A single vertex may be considered a 0-simplex, and a line segment may be considered as a 1-simplex, a 2-simplex is a triangle and a 3-simplex is a tetrahedron (Lindemann 2009), as shown in figure(5).



Figure(5) Simplex types

In our paper, GJK algorithm for calculating the shortest space between a pair of 2D convex polygon uses a 1-simplex and 2-simplex respectively and must be the closest to origin point (Wikipedia 2013).

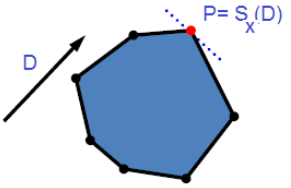
4.3. Support Mapping Function

The quickness of GJK algorithm results from the fact that it relies solely on support mappings to completely describing the geometrical shapes. Actually instead of explicitly computing the set of Minkowski difference we will construct it implicitly by using the support function $S_{X,Y}$ to reduce the running time of GJK algorithm (Wikipedia 2013). Clearly, the purpose of support function for different types of convex shapes is to build a simplex polygon inside Minkowski differences and this can be done by choosing the farthest point according to direction vector D of shape as indicated in equation(2) and figure(6). Then flip or reflect the direction vector -D to find the extreme point for second shape in that direction.

$$S_X(D) = \{p : \max\{D \cdot a : a \in X\} = v \cdot p, p \in X\} \quad \ldots(2)$$

The support point results from subtraction of the two farthest points of shapes as in equation (3) use to terminate the algorithm or to find new direction vector (Lindemann 2009), (Jovanoski 2008), (Olvång 2010).

$$S_{X \ominus Y} = S_X(D) \ominus S_Y(-D) \quad \ldots(3)$$



Figure(6) farthest point on convex hull

4.4. Closest Point on Line Segment

GJK distance algorithm finds out the Euclidian distance between the pair of convex objects which can be calculated by finding the closest vertex on the simplex polygon to the origin point. Simplex for 2D space could be either a single point C or line segment AB. The problem in this section is to determine the point C on AB closest to origin then the distance is the magnitude of the closest point (Onderik 2013). Actually, the basic approach for determining such a vertex is that a closest point can only be a vertex of the target polygon, or the projecting of origin point onto the extended line through AB. Check if the projection point C lies within the line (Diego 2010). Then, C is the correct answer. Obviously, any point lies within line AB can be expressed using parametric equation as in (6).

$$AB = B.x - A.x, B.y - A.y \quad \ldots(4)$$

$$A0 = 0 - A.x, 0 - A.y \quad \ldots(5)$$

$$C(t) = A + t(AB) \quad \ldots(6)$$

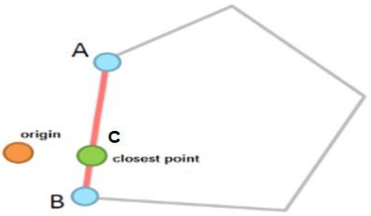
and by using the important properties of the dot product operation. Then, t resultant to the projection of origin point onto the line AB as in equation (7).

$$t = (A0) \cdot \frac{n}{\|AB\|} \quad \ldots(7)$$

Where t value must in interval [0..1] and n is the unit vector in the direction of line AB calculated using equation(8).

$$n = \frac{AB}{\|AB\|} \quad \ldots(8)$$

Figure (7) show the closest point C on the line AB to the origin point (Catto 2011).



Figure(7) Closest point on simplex

Closest Point on Line Segment Algorithm

Input: The start point(A.x , A.y) and end point (B.x , B.y) of line segment.

Output: Closest point C on line AB segment

Begin

Step1: Create AB line segment and A0 line using equation (4)and (5) respectively.

Step2: Project A0 on to line AB.

Step3: Compute squared of length | | AB| | by

$$(AB.x \times AB.x) + (AB.y \times AB.y)$$

then apply equation (8). To obtain the unit vector.

Step4: Compute distance t along AB line using equation (7).

Step5: Calculate the closest point C using equation (6)

Step6: Return C.

Finish

5. Overview of GJK Distance Algorithm

The enhanced Gilbert-Johnson-Keerthi distance algorithm provides an iterative method to determine the Euclidian distance between two convex sets X and Y presented by Stephen Cameron in 1997 then, it cited dramatic improvement especially with high dimensional convex hull such as it approached the problem in geometrical rather than algebraic method. This make GJK algorithm more simple and robust in implementation. Basically, GJK algorithm uses the Minkowski difference and support mapping function to compute the

shortest distance of the origin point to the simplex shape with linear complexity by creating and iteratively updating the simplex inside the Minkowski difference. Actually in each iteration, GJK builds and checks a simplex within Minkowski difference vertices that lies nearer to the origin point than the simplex shape building in the previous iteration (Lindemann 2009). If the origin point locates outside the convex of Minkowski difference then the closest point will be on the boundary of it; otherwise, the closest point will be the origin point itself. The first termination condition for the iteration of GJK algorithm is to check wither the computed closest point is the origin then the algorithm finished with distance 0. This means that the two convex objects were collided, while the second termination condition is to check if the current closest point was founded in the previous iterations of the algorithm. Then, the algorithm is also terminated with 0 distance. The third termination condition is to check if the distance of a new obtaining Minkowski Difference point with current points is less some value called tolerance which assumes to be near of the zero value (Catto 2011).

GJK Distance Algorithm (Code Zealot 2010)
Input: Two convex hull objects X and Y.
Output: Shortest directed distance from Y to X.
Begin
Step1: Initialization step which involved the following:

- Set the simplex array S=Ø.
- Set the number of simplex vertices k=0.
- Set the direction of the vector D(1 , -1).

Step2: Compute the support point $S_{X\ominus Y}(D)$ using equation (3) and add it to
simplex S[k++].
Step3: Flip the direction vector to –D to compute the second support point.

Step4: Compute the second Minkowski Difference point using support function

$S_{XeY}(-D)$ and add it to simplex S[k++].

Step5: While (true)

Step6: Begin

Step7: Search of point C on the current simplex closest to the origin using the

above closest point on line segment algorithm and check the first

termination case if so, then return false.

Step8: Obtain the new direction vector D as follow:

- Flip or negate C.
- Compute C magnitude using equation(9).

$$magnitude = \sqrt{C.x^2 + C.y^2} \quad \ldots(9)$$

- Normalized C by using equation(10)

$$C.x = C.x/magnitude$$
$$C.y = C.y/magnitude \quad \ldots(10)$$

- Set D = C.

Step9: Calculate the new Minkowski Difference point Z along a new direction D.

Step10: Check the second termination case.

Step11: Check if the new Minkowski Difference point far enough along D by using

dot product operation with it. put the result in DZ.

Step12: Check if the simplex point X far enough along D by using dot Product

operation. Put the result in DX.

Step13: Check the third termination case by DZ-DX<tolerance then return DZ as

final distance value.

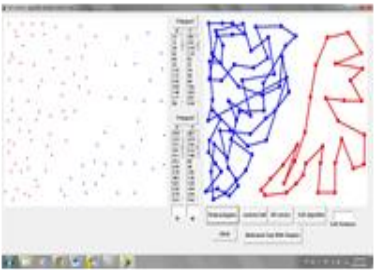
Step14: discard one of simplex points X or Y and keep the closer point to origin.

Step15: end for while loop.

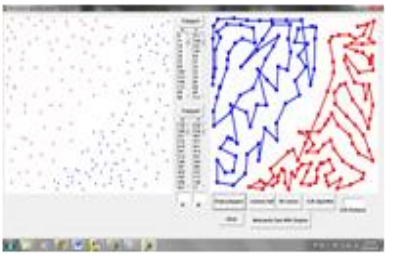
Finish

6. Experimental Results and Implementation of System Software

In this section, more details can be found of system software implementation. We explain the results of software implementation with two examples of points. First example shows not collide point sets of two convex polygons, while second example shows the collision detection problem between them. The first step of system begins by generating two sets of randomly points on 2D space, each set elected to construct one of convex hull polygons. Actually any user of system could utilize the mouse move and mouse down operations to generate of 2D points. Figure (8) depicted the generation operation of two 2D points set and their intersected and not intersected polygons.



Figure(8.a) GUI of not collided points set

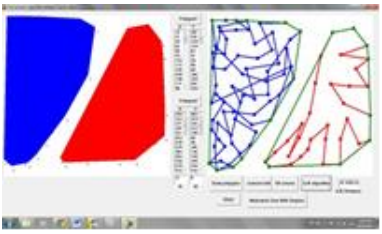


Figure(8.b) GUI of collided points set

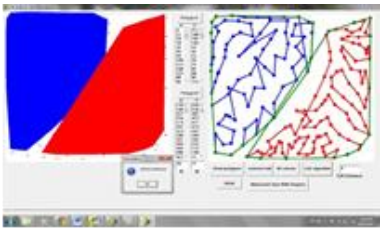
Figure(8) the GUI of randomly generated points for two sets

After having the 2D points for each polygon we sort these points relatively to x-coordinate value and then, applying Andrew's Monotone Chain algorithm to construct their convex hull. Basically, from our implementation of system we could see that the Monotone Chain algorithm could deal with any number of polygon points in the same effectiveness since it splits them into upper and lower hulls and then construct one convex hull in O(nlogn) time complexity. Finally, we apply GJK

algorithm to compute the minimum distance between these two convex polygons depending on the ability to build the simplex polygon in Minkowski differences and determine the closest point in the simplex to origin point. If there is collision detection problem then the closest point is origin point itself and GJK finished with zero distance as we could be see in figure (9).



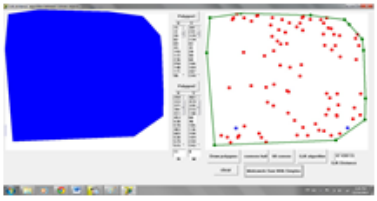
Figure(9.a) minimum Distance with about 47 pixel



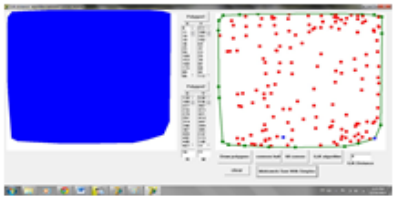
Figure(9.b)Collision Detection with

Figure(9)The GUI of Shortest Distance between two convex hull

Figure (10) show the GUI of two collided and not collided convex hulls with their Minkowski difference and line segment point of simplex in blue colour of the convex of Minkowski difference. We could see that the line points might be points in the boundary or inside of Minkowski difference and this depends on the initial value of Direction vector.



Figure(10.a)not collided Minkowski Difference shapes



Figure(10.b) collided Minkowski Difference shapes

Figure(10) Minkowski Difference and line segment simplex

7. Conclusion

In this paper Gilbert-Johnson-Keerthi's procedure was presented to indicate of collision detection problem between two convex objects in 2D space and the closest point between line segment and origin point. Despite the age of GJK algorithm with about 20 years, the proposed algorithm is very convenient for calculating the shortest distance between shapes, because along with its simplicity it offers high enough speed with linear time complexity. Actually, the mathematical background makes this algorithm fast and versatile when applied to solve the problem. Also, it can handle many types of shapes wither in 2D or 3D space. Moreover, GKJ algorithm used in popular and widely applications such as real time collision detection routine, proximity queries and path planning. The algorithm was implemented in Delphi programming language and tested on a PC under Microsoft Windows operating system.

BIBLIOGRAPHY:

Bernabeu, E.J., A. Valera, J. Gomez-Moreno. 2013. "Distance Computation between Non-Holonomic Motions with Constant Accelerations." *International Journal of Advanced Robotic Systems.*

Catto, E. 2011. "Computing Distance." Blizzard Entertainment available at http://twvideo01.ubm-us.net/o1/vault/gdc10/slides/Catto_Erin_PhysicsForProgrammers_ComputingDistance.pdf

CGAL. 2013. "2D Minkowski Sums." accessed November 24, http://doc.cgal.org/latest/Minkowski_sum_2/index.html

Code Zealot. 2010. "*GJK – Distance & Closest Points Code Zealot.*" accessed December 13, http://www.codezealot.org/archives/153.

Diego M.J. 2010. "Closest Point on line segment to point." accessed on Aug 06, available at http://www.slideshare.net/bonbon889/closest-point-on-line-segment-to-point.

GeomAlgorithms. 2012. "The Convex Hull of a 2D Point Set or Polygon", accessed November 13, www.softsurfer.com/Archive/algorithm_0109/algorithm_0109.htm

Gilbert, E. G. and C.-P. Foo. 1990. "Computing the distance between general convex objects in three-dimensional space." In *IEEE Transactions on Robotics and Automation* 6(1): 53–61.

Gilbert, E. G., D. W. Johnson, and S. S. Keerthi. 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space." *IEEE Trans. Robot. Autom* 4(2):193–203.

Jovanoski, D. 2008. "The Gilbert – Johnson – Keerthi (GJK)Algorithm." available at http://www.cosy.sbg.ac.at/~held/teaching/bakk_seminar/se_arbeiten_07-08/KollisionserkennungGJK.pdf , Department of Computer Science University of Salzburg.

Lindemann, P. 2009. "The Gilbert-Johnson-Keerthi Distance Algorithm." University of Munich, Germany, the *Media Informatics Proseminar on Algorithms in Media Informatics.*

Liu, J.S., Yr. Chien, S.P. Shiang, W-C. Lee. 2001. "Geometric Interpretation and Comparisons of Enhancements of GJK Algorithm for Computing Euclidean Distance between Convex Polyhedra." Institute of Information Science 20 Academia Sinica Nankang, Taipei 115, Taiwan.

Olvång, L. 2010. *Real-time Collision Detection with Implicit Objects.* Master Thesis, Institutional for information

technology, Department of Information Technology. Uppsala University.

Onderik, J. 2013. "Narrow phase Collision Detection." available at www.sccg.sk/~durikovic/classes/CGAnim/ca10_lesson06.pdf.

PAVZAV. 2010. "Andrew's Monotone Chain Convex Hull Algorithm." accessed November 20, http://pavzav.blogspot.com/2010/12/andrews-monotone-chain-convex-hull.html.

Sacristan, V. 2012. "CONVEX HULL IN 2D", available at www-ma2.upc.es/vera/wp-content/uploads/2012/10/CH-2D.pdf_

Senin, M., N. Kojekine, V. Savchenko, I. Hagiwara. 2003. "Particle-based Collision Detection." International conference on computer graphics *The Eurographics Association,* Granada, Spain.

Wang, W., S. Gong, M. Chui, P. Li. 2012. "Research on Convex Polyhedron Collision Detection Algorithm Based on Improved Particle Swarm Optimization." International Conference on Mechanical Engineering and Material Science, China 2012.

Twisted Oak Studios. 2012. "Minkowski sums and differences." accessed November 17, http://twistedoakstudios.com/blog/Post554_minkowski-sums-and-differences.

Wikipedia, Incnis Mrsi. 2013. "standard simplex." accessed December 3, http://en.wikipedia.org/wiki/Simplex.