Journal of Kerbala University , Vol. 9 No.2 Scientific . 2011

Implementation of Calculating the area and centeriod of two dimension Convex hull using " Andrews variant of the Graham scan" algorithm

م. آسيا مهدي ناصر حسين الزبيدي
جامعة كربلاء\ كلية العلوم / قسم الحاسبات

Chapter one
Calculating the area and centeriod of two dimension Convex hull

المستخلص:
إنَّ الشكل الهندسي المضلع(التل المحدب) والمسمى (Convex Hull) هو أحد الأشكال الهندسية المهمة والذي يستخدم في انشاء أشكال هندسية أخرى. في هذا المشروع استخدمنا خوارزمية (Andrew's variant of the Graham Scan) للحصول على الشكل المحدب او المضلع (Convex Hull) وذلك عن طريق حذف كل النقاط الداخلية واحاطة(bounding) النقاط الخارجية التي تؤلف الشكل المحدب فقط , والهدف الاساس من البحث هو حساب مساحة ومركز اي شكل مضلع .

Abstract

A convex Hull is an important structure in graphic geometry that can be used in the construction of many other geometric structures. In this paper we used "Andrew's variant of the Graham's Scan algorithm" to obtain the Convex Hull by eliminating all the interior points and bounding box the outer points that construct the convex hull , the main object of this paper is to compute the area and centroid of any convex Hull.

1. Introduction:

The aim of this paper is to compute the Convex Hull which is a cycle graph whose vertices are composed of all the points inside it and all the interior angles in it are less than 180 degree, and then calculating the area and centroid of the Convex Hull which is useful in the following applications: [1]

- Collision detection in video games, providing a useful replacement for bounding boxes.
- Hidden object determination.
- Shape analysis.
- Visual pattern matching/ object detection.
- Mapping.
- Path determination.

2. Building of two dimension Convex Hull: A polygon is closed plan figure with n sides, polygon derived from the Greek "poly" meaning many and "gonia" which means angle. The most familiar type of polygon is the convex polygon also called Convex Hull or just Hull. To understand what is Convex Hull we first know what is the Non Convex Hull [2].

(2-1) Non Convex Hull: There are many ways to draw a boundary around a set of points in a two-dimensional plane; one of the easiest ways to implement this is a bounding-box manner, which is a rectangle that spans the set from its minimum and maximum points in the X and Y planes. Creating a bounding-box is easy, but it doesn't form as tight a wrapper as we might like around a set of points, So it's Non Convex Hull [3]. while if it become more tight using straight lines and with direct connect between external points that may called convex Hull
Example: Consider that we have a set of points S with two points P & Q if the line PQ is outside the convex even part of it then we will get a Non Convex Hull and vice versa as shown in figure (1).

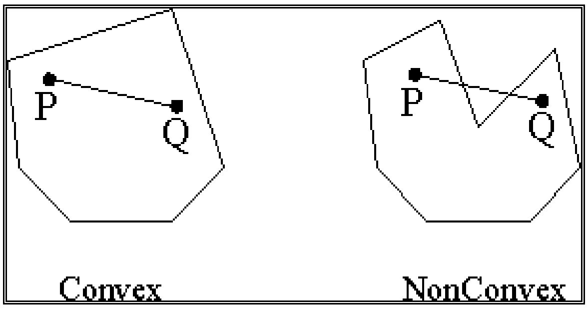


Figure (1) convex & non convex hull

that mean may one at least of the interior angles of the Non Convex Hull more than 180 degree[4].
(2-2) Two dimension Convex Hull: According to what we have mentioned before about the Non Convex polygon, the Convex Hull should have the following properties:

1. If we have a set of points S then all the vertices that can be drawn between any two points in it should be inside the Convex Hull.
2. No point in set S lies outside the Convex Hull.
3. All the interior angles in the Convex Hull are less than 180 degrees. figure (2) show an example of Convex Hull

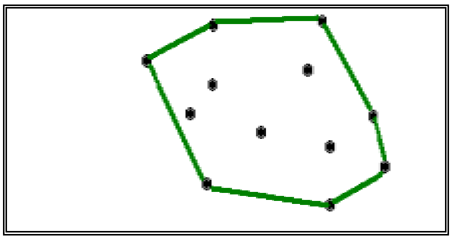


Figure (2) Convex Hull polygon

Computing a convex hull (or just "hull") is one of the first sophisticated geometry algorithms, and there are many variations of it, The most common form of geometry algorithm involves determining the smallest convex set (called the "convex hull") containing a discrete set of points, This algorithm also applies to a polygon, or just a set of segments, whose hull is the same as the hull of its vertex point set [5].

3. Andrew's variant of Graham Scan Algorithm: The algorithm I'll demonstrate here is referred to as the Andrew's variant of the Graham scan, is a method of computing the convex hull of a finite set of points in the plane with time complexity of O(n log n) time, and Andrews variation is a simplification that requires a bit less computation. It is named after Ronald Graham. The algorithm finds all vertices of the convex hull ordered along its boundary, and here is the algorithm steps and will explain each step in more detail. [6]

A. generate a set of points randomly using a specific function (Random) which return the X and Y coordinates for each point according to the object coordinates and then store them in array of Tpoint including X and Y values.
B. Sorting the array of points according to X-axis in non decreasing order using bubble sort algorithm.
C. Select extremes points:

- Select left extreme point as the left most point which is the first point in the sorted array using the equations:
  Left point.x = sorted array.x [0] ------------ (1)
  Left point.y = sorted array.y [0]
- Select right extreme point which is the last point in the sorted array using the equation:
  Right point.x = sorted array.x [n-1] ----------- (2)
  Right point.y = sorted array.y [n-1]
- Remove left and right extreme points from the array.

D. Draw line from left point to right point to use it in determining the upper and lower half.

    If the point is above the line
        Add the point to the end of upper array
    Else
        Add the point to the end of lower array

Then Sorting the remaining points into the upper and lower sets requires that we have some function that determines whether a point is above or below a line. We accomplish this using the following strategy: Given a set of points on a line, p0, p1, and p2, we first perform a coordinate translation so that p1 is at 0,0. we then take the determinant of p0 and p2. The resulting value will be negative if p2 angled off in the left direction, positive if it has moved to the right, and 0 if it is collinear with the first two points, The equation that is used to get this determinant looks like this:

Determination= ((p0 x-p1x) *(p2y-p1y)) - ((p2x-p1x) *(p0y-p1y)) ----(3)

Partitioning set S into upper and lower is simply a matter of iterating over each point, calculating the determinant, and moving it into lower for a value ≥ 0, or upper for a value < 0.

E. Construct the upper and lower Hull: Now that points are partitioned, the actual construction of the hull can begin. From the algorithm description, we can see that the upper and lower hull construction steps are symmetrical. Both proceed from right to left, starting at the leftmost point and moving to the rightmost point. The only difference is the source of their input points, and the direction they check to insure convexity. the upper or lower hull is started by simply adding left to the output hull, Points are then added from the correct input source. As each point is added, if the number of points in the working hull is equal to 3 or more, a test is made to see if the last three points have created a convex angle. Testing for the convex angle is done using the same determination formula as in equation (3). If the hull has n points, we simply test to see if Pn-1 is above or below the line formed by Pn-2 and Pn. When constructing the lower hull, if we see that the point is above the line, we have violated convexity, and the middle point is removed from the hull. The opposite test is made when constructing the upper hull. This test-and-remove process is repeated until the last three points are convex, or there are fewer than 3 points in the working hull as shown below:[7]

Construct the lower hull

- Add left to lower_hull.
- While lower is not empty.
- add lower[0] to the end of lower_hull.
- remove lower[0] from lower
- while size(lower_hull >= 3) and (the last 3 points lower_hull are not convex)
- remove the next to last element from *lower_hull*

Construct the upper hull

- Add left to upper_hull
- While upper is not empty
- add upper[0] to the end of upper_hull
- remove upper[0] from upper
- while size(upper_hull >= 3) and (the last 3 points upper_hull are not convex)
- remove the next to last element from *upper_hull*. [8]

F. Merge the upper and lower Hull: The final step, merging the upper and lower hulls, is a append one hull to the other, and removing the extra copy of right and left points. Once that is done, the actual convex hull definition can be given as a list of points, starting with left and moving counter-clockwise around the hull, constructing the complete hull. [9]

4. Area of two dimension Convex Hull: After getting the Convex Hull now we will compute the area of it, after many attempt and research in computing the area of the Hull (polygon) we fined that the easiest and the most efficient way to do that by dividing the polygon into triangles and calculate the area of each one of them, finally with summation of them we will have the area of the Whole polygon (Convex Hull). Consider a polygon made up of line segments between N vertices (Xi,Yi), i=0 to N-1. The last vertex (Xn,Yn) is assumed to be the same as the first, (the polygon is closed as shown in figure (3). [10]

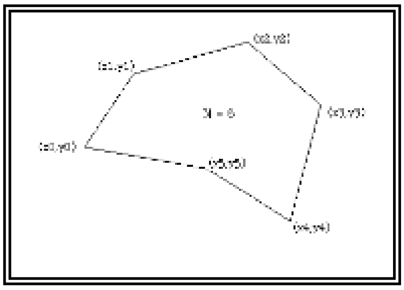


Figure (3) polygon of six vertices

The area given by the equation:

$$A = \frac{1}{2} \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i) \quad \text{-------- (4)}$$

The sign of the area expression above (without the absolute value) can be used to determine the ordering of the vertices of the polygon. If the sign is positive then the polygon vertices are ordered counter clockwise about the normal, otherwise clockwise.[11]

4. Convex Hull Centroid: The centroid is also known as the "center of gravity" or the "center of mass". The position of the centroid assuming the polygon to be made of a material of uniform density is given below. As in the calculation of the area above, the polygon is closed.[12],The centroid is given by the equations:

$$c_x = \frac{1}{6A} \sum_{i=0}^{N-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$c_y = \frac{1}{6A} \sum_{i=0}^{N-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \quad \text{--------- (5)}$$

Chapter two
Practical constructed of two – dimension Convex Hull

Practical constructed of Convex Hull: The algorithm starts with an unordered set of points defined by Cartesian coordinates - each point has a position on the X-axis and Y-axis [11]. we explain each step in more detail practically with example.

Andrew's variant of the Graham scan algorithm:
Input: array of points.
Output; Array of Convex Hull polygon points and then computes area and centroid of it.
Begin

Step1: generate N vertices (points) randomly by using special function (Random) and store the points in array of Tpoint (which is a type of structure) including X and Y axis as shown in figure (4) when n (number of vertices of convex hull) equal to 50

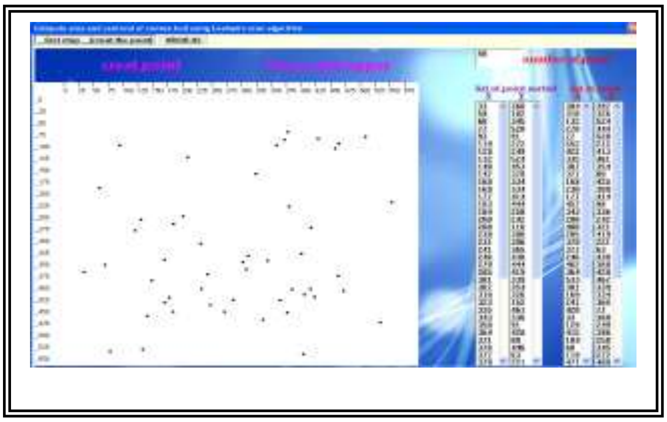


Figure (4) 50 points after and before sorting

Step2: Choose of extreme points
2.1 Choose the leftmost point (left extreme point)
2.2 Choose the right most point (right extreme point)
2.3 Draw line between the leftmost and the rightmost
2.4 Remove left and right from the sorted array

Step3:
3.1 while there are still points in the sorted array
3.1.1 Remove point from it
3.1.2 If the point is above the line
3.1.3 Add the point to the end of upper array
Else:
3.2 Add the point to the end of lower array. As shown in figure (5) .

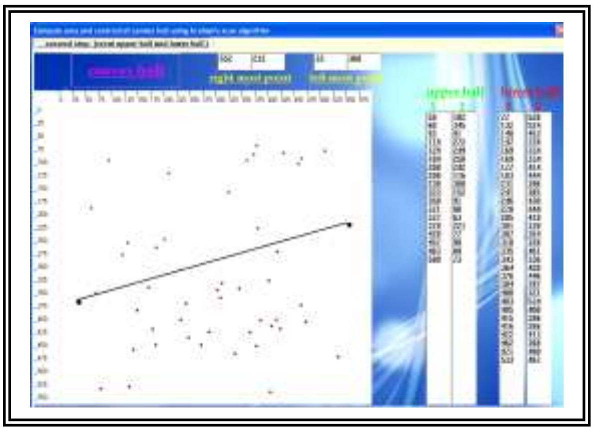
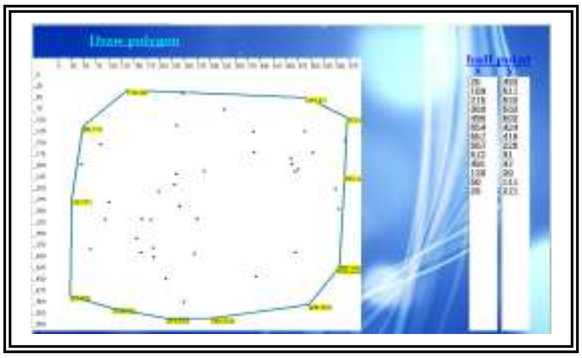


Figure (5) determine the upper and lower halves

Step4:
4.1 Construct lower Hull
4.1.1 Add left to lower_hull
4.1.2 While lower is not empty
4.1.3 Add lower[0] to the end of lower_hull
4.1.4 Remove lower[0] from lower
4.1.5 while size(lower_hull >= 3 and the last 3 points lower_hull are not convex
4.1.6 Remove the next to last element from *lower_hull* as equation (3)
4.2 Construct the upper hull
4.2.1 Add left to upper_hull
4.2.2 While upper is not empty
4.2.3 Add upper[0] to the end of upper_hull
4.2.4 Remove upper[0] from upper
4.2.5 while size(upper_hull >= 3 and the last 3 points upper_hull are not convex
4.2.6 Remove the next to last element from upper_hull. As shown in figure(6).



Figure(6) convex hull of 50 points

Step5: Compute area
Step6: Compute centroid. As shown in figure (7)
Step 7: end or finish the algorithm.

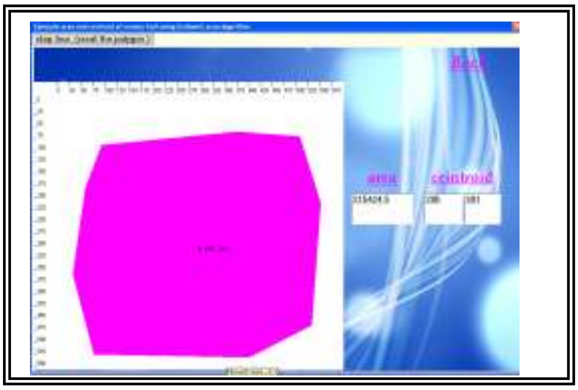


Figure (7) area and centriod of convex hull

Conclusions

1- The number of points in the Convex Hull even large or small will not effect to having the Convex Hull also the area and centroid of it.
2- The Random function which is used in generating two dimension points is a speedy function in having random points.
3- The time complexity O(n log n) in the Incremental Graham algorithm coming from the way of sorting the points and dividing the Convex Hull into upper and lower Hull.
4- The time complexity O(n log n) is the fastest in corporation between points.
5- There is no other ways in computing the area and centroid of the Convex Hull except by dividing it into triangles.

Future works:

1- Generate and use points in having three dimensions Convex Hull.
2- Apply the area and centroid of the Convex in applications.
3- Show the ability of other algorithms and compare between them.

References

[1] Dr.Dobb's Portal, "World of the Software Development", September, 2007.
[2] Mark Nelson, "Building the Convex Hull", August, 22nd, 2007.
[3] The Mathworld classroom, "Polygon",2002.
[4] S.G. Akl & Godfried Toussaint, "Efficient Convex Hull Algorithms for Pattern Recognition Applications", Proc. 4th Int'l Joint Conf. on Pattern Recognition, Kyoto, Japan, 483-487 (1978).
[5] A.M. Andrew, "Another Efficient Algorithm for Convex Hulls in Two Dimensions", Info. Proc. Letters 9, 216-219 (1979).
[6] A. Bykat,"Convex Hull of a Finite Set of Points in Two Dimensions", Info.Proc. Letters 7, (1978).
[7] En.wikipedia.org, "Ronald Graham".
[8] W. Eddy,"A New Convex Hull Algorithm for Planar Sets", ACM Trans. Math. Software 3(4), (1977).
[9] Ronald Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Point Set", Info. Proc. Letters 1, 132-133 (1972).
[10] Paul Bourke, "calculating the area and centroid of a polygon" July, 1988.
[11] Schere Peter "Area inside Convex Hull" Wedm 4th, April, 2001.
[12] Franco Preparata & S.J. Hong, "Convex Hulls of Finite Sets of Points in Two and Three Dimensions", Comm. ACM 20, 87-93 (1977).